Wrocław University of Science and Technology
Faculty of Information and Communication Technology

Field of study: **Applied Computer Science (IST)**

MASTER THESIS

Comparison of Snowflake and Databricks platforms in the context of data processing and business analytics

Damian Pokrywka

Supervisor

Dr inż. Bogumiła Hnatkowska

Keywords: Data Warehouse, Data Lake, Data Lakehouse, Snowflake, Databricks, Cloud Engineering

WROCŁAW 2024

ABSTRACT

In the current area of cloud-based data management, Snowflake and Databricks have gained reputation as leading platforms for data warehousing and data engineering. However, a clear comparison to determine the better platform for specific tasks is often lacking. This thesis aims to address this gap by providing a detailed comparison of Snowflake and Databricks to guide organizations in their decision-making process. The primary goal of this thesis is to evaluate the strengths and weaknesses of both platforms. This evaluation was conducted through an analysis of official documentation, related works and practical experiments based on a reference architecture. A summary of the conclusions from the analysis and experiments can be found at the end of the thesis. In conclusion, this thesis provides valuable insights into the comparative benefits of Snowflake and Databricks. It offers guidance for organizations to make informed decisions based on their unique data requirements and strategic goals, emphasizing the importance of aligning platform capabilities with business needs to maximize value from cloud-based data solutions.

STRESZCZENIE

W obecnym obszarze zarządzania danymi w chmurze, Snowflake i Databricks zdobyły reputację wiodących platform do hurtowni i inżynierii danych. Jednak często brakuje jasnego porównania, które pomogłoby określić lepszą platformę do konkretnych zadań. Niniejsza praca ma na celu wypełnienie tej luki, dostarczając szczegółowe porównanie Snowflake i Databricks, aby wspomóc organizacje w procesie podejmowania decyzji. Głównym celem tej pracy jest ocena mocnych i słabych stron obu pltatform. Ocena ta została przeprowadzona poprzez analizę oficjalnej dokumentacji, powiązanych prac oraz praktycznych eksperymentów bazując na referencyjnej architekturze. Podsumowanie wniosków z analizy i eksperymentów znajduje się na końcu pracy. Podsumowując, niniejsza praca dostarcza cennych spostrzeżeń na temat porównawczych korzyści Snowflake i Databricks. Oferuje wskazówki dla organizacji, aby podejmowały świadome decyzje na podstawie swoich unikalnych wymagań dotyczących danych i celów strategicznych, podkreślając znaczenie dopasowania możliwości platformy do potrzeb biznesowych w celu maksymalizacji wartości z rozwiązań chmurowych.

CONTENTS

INTRODUCTION

In the era of big data and advanced analytics, organizations are increasingly relying on sophisticated data platforms to manage, process, and derive insights from their data. Two leading platforms in this domain are Snowflake and Databricks, both of which offer unique capabilities adapted to meet the demands of modern data engineering and business analytics. As companies aim to leverage their data effectively, understanding the strengths and limitations of these platforms is essential for informed decision-making. Despite the increasing adoption of Snowflake and Databricks, there is a observable lack of comprehensive relative studies evaluating these platforms based on real-world use cases and reference architectures. This knowledge gap presents a significant challenge for organizations trying to select the most appropriate platform for their specific requirements.

In the first chapter, there will be fundamental information about data warehouse, lake, and lakehouse concepts, as well as an overview of Databricks and Snowflake. At the end of the chapter, a reference architecture will be presented, which will be used for the comparison of these data platforms.

In Chapter 2, there is a review and analysis of related works pertinent to this thesis and a description of the methodologies used in these articles that can be applied in the analysis and experiments in this thesis.

Chapters 3 and 4 are directly related to the reference architecture and are divided into sections corresponding to the components in the architecture. The structure of these chapters is to first analyze the available tools and then use this information to make the comparison and draw conclusions.

At the end, there is a short chapter 5 about the user experience during the analysis and experiments conducted for this thesis.

AIM OF THE THESIS

The aim of this thesis is to conduct a comprehensive comparison of Snowflake and Databricks based on a reference architecture used by a major IT company. This thesis will begin with a bacground and review of related works, including scholarly articles, industry reports, and case studies, to establish a solid foundation of existing knowledge and methodologies. Following this review, the thesis will dive in a detailed analysis of the available tools and solutions provided by both Snowflake and Databricks. To provide a practical perspective, the thesis will also include hands-on experiments with AI tools in

cloud data platforms. These experiments will be designed to simulate real-world scenarios and evaluate the performance and costs of AI-driven data processing and analytics on both Snowflake and Databricks. By combining theoretical insights with empirical data, this study aims to provide a comparison that can assist IT companies in choosing the most suitable platform for their specific needs.

SCOPE OF THE THESIS

The scope of this thesis is to explain the fundamental concepts of data cloud platforms and review related works to analyze the methodologies that can be used in the thesis. Additionally, it is necessary to analyze available tools for the aspects in the reference architecture and, based on this analysis, prepare corresponding research and conclusions of the comparison. Finally, the thesis will include a conclusion that can be useful for companies in determining whether Databricks or Snowflake is better for specific aspects in the future. A reference architecture, introduced at the end of the first chapter, serves as the basis for comparison. Chapter 2 reviews related works and methodologies from scientific articles, industry reports, and case studies, establishing a foundation of existing knowledge. Chapters 3 and 4 analyze the tools related to components of the reference architecture, followed by a detailed comparison of Snowflake and Databricks, highlighting their strengths and weaknesses. Chapter 5 discusses user experiences during the analysis and experiments, offering practical insights. The aim is to assist IT companies in choosing between Snowflake and Databricks by combining theoretical insights with empirical data from hands-on experiments.

1. BACKGROUND

1.1. DATA WAREHOUSE

Data warehouses have become a crucial part of modern decision making system in many organizations. These systems offer effective access to historic and combined data from different sources to support management people with decision making processes. Data warehouses have been growing over time, and recent years have seen new solutions that enhance their scalability and flexibility, allowing them to support a wider range of data types and workloads [6] [14]. Modern data warehouses have several key features that can be found in literature [6]:

— **Multi-Tenant Architecture:** data warehousing uses a multi-tenant architecture to enable simultaneous access for multiple users. Sharing data and working together on analytics projects is made better with this solution.
— **Cloud-Based Infrastructure**: data warehouses are becoming more and more popular because of their scalability, flexibility, and cost-effectiveness. Using cloud-based infrastructure allows organizations to easily scale up or down their data volume and processing requirements.
— **Advanced Analytics Capabilities:** data warehousing is a type of warehousing that employs advanced analytics techniques such as machine learning, natural language processing, and predictive analytics. By using these tools, organizations can gain deeper insights from their data, leading to more informed decision-making.
— **Support for Structured and Unstructured Data:** both structured and unstructured data, including text, images, audio, and video, can be stored in modern data warehouses. This simplifies the process of integrating and analyzing data for organizations from a variety of sources.

Modern data warehouse typically consists of three layers [6]:

— **Data Storage Layer:** archiving information is the responsibility of the storage layer of the data warehouse. The implementation of this layer is common in modern data warehousing by means of a distributed file system e.g. Amazon S3.
— **Data Processing Layer:** the data processing layer handles the processing of data from the data warehouse. This layer usually uses cloud solutions for data processing, e.g. Amazon Glue.

— **Data Access Layer:** users are granted access to data stored within the data warehouse by the data access layer. The implementation of this layer in modern data warehouse setups is usually done by using business intelligence tools or by using data science platforms e.g. Databricks.

The figure 1.1 below provides a comprehensive overview of the modern data warehouse architecture.

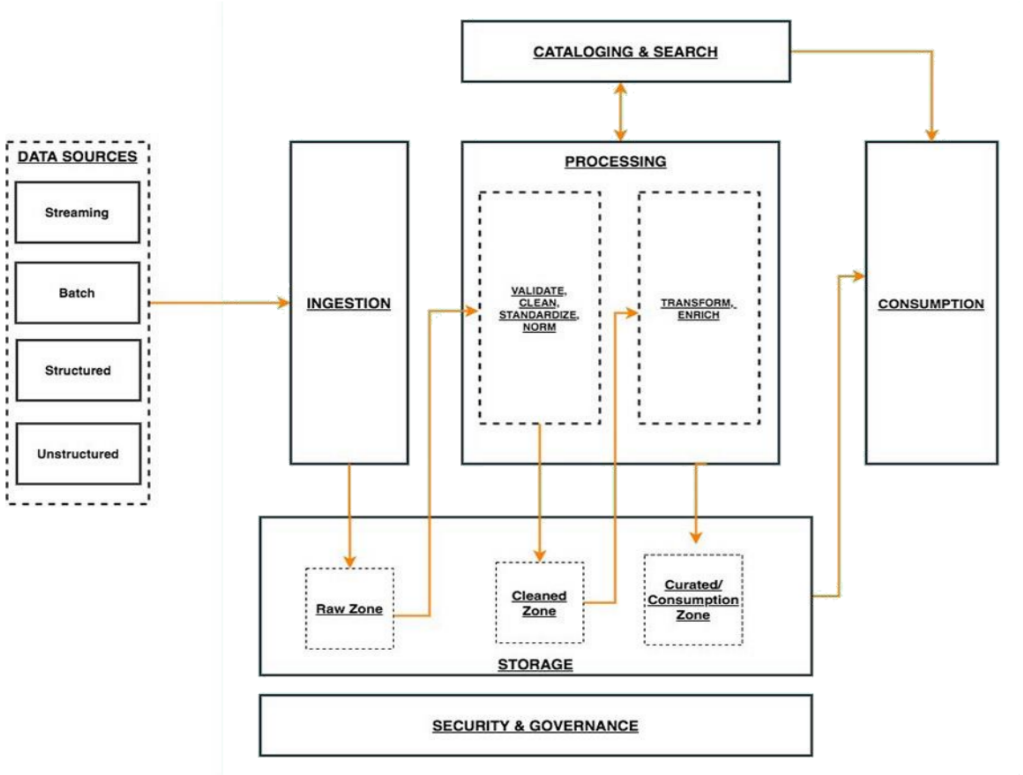


Fig. 1.1: Data warehouse architecture [6]

A data warehouse transforms unstructured data into a structured database to facilitate analysis and reporting. Online Transaction Processing (OLTP) systems are responsible for monitoring standard business activities like sales, purchases, and inventory adjustments. The typical phases in data warehouse involves the extraction, consolidation, filtering, transformation, cleansing, aggregation, and loading of raw data into the data warehouse, ensuring that different formats are seamlessly integrated. The journey of information from its primary sources to the data warehouse is depicted in the below diagram (Fig. 1.2).

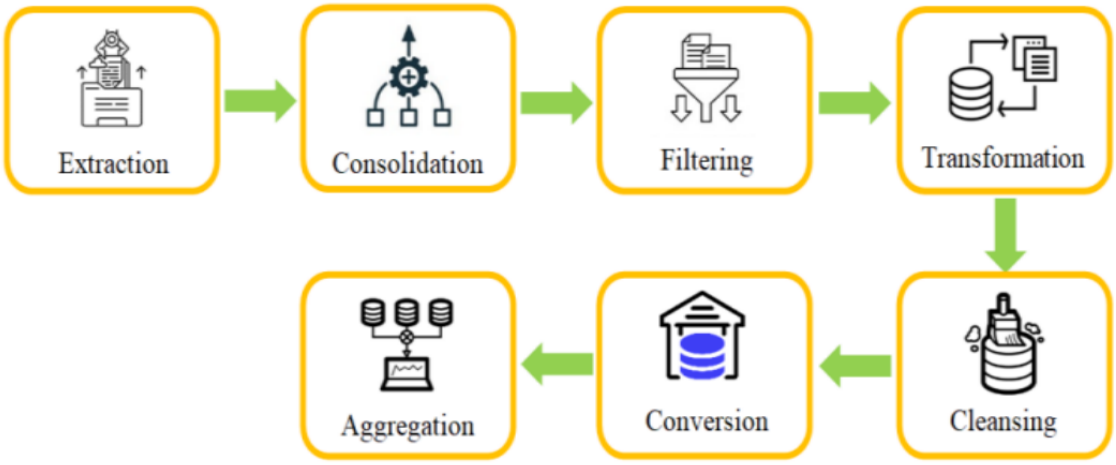


Fig. 1.2: Process of transforming raw data into a data warehouse [6]

Below is a description of the phases from the diagram (Fig. 1.2) [6]:

— **Extraction:** this phase typically involves working with multiple data sets. Data is extracted from various sources and transformed into a format suitable for the data warehouse setup.
— **Consolidation:** this method combines various data sets into a single repository, creating a unified data source.
— **Transformation:** during this phase, data undergoes a series of transformations to derive new values for the target system or to verify the source data.
— **Cleansing:** this crucial step ensures the reliability of data by addressing inconsistencies in field lengths, descriptions, value assignments, missing entries, and integrity constraint violations, making the data in the warehouse more accurate.
— **Conversion:** this step involves typing and translating raw data into the format used by the warehouse and mapping it onto new data fields within the warehouse.
— **Aggregation:** in this final stage, data is classified and connected into usable metrics for analysis, which significantly enhances the value of the data warehouse.

1.2. DATA LAKE

By the beginning of the 21st century, new types of diverse data were emerging in ever-increasing volumes on the Internet and at its interface with enterprises (e.g., web-based business transactions, real-time streaming, sensor data, and social media). With this huge amount of data, the need for better solutions for storing and analyzing large volumes of semi-structured and unstructured data to gain relevant information and valuable insights became apparent. Traditional schema-on-write approaches, such as the extract, transform, and load (ETL) process, proved to be inefficient for such data management requirements.

This led to the rise of another popular modern enterprise data management scheme known as data lakes [18].

A data lake is a central repository designed to store big amounts of data in its original, unprocessed state. Unlike hierarchical data warehouses that organize data in files and folders, data lakes use a flat architecture and object storage system. This system uses metadata tags and unique identifiers to improve data retrieval and location efficiency across different regions, thereby improving performance. By utilizing cost-effective object storage and open formats, data lakes facilitate efficient data usage across various applications [5].

Data lakes were designed to address the weakness of data warehouses. Although data warehouses provide high-performance and scalable analytics, they are expensive, proprietary, and not suitable for many modern applications. Data lakes unify an organization's data in a single, central repository, allowing it to be stored in its original, raw state without requiring a predefined schema. This allows the storage of raw data, structured data from sources like database tables, and intermediate data produced during processing. Unlike traditional databases and data warehouses, data lakes can handle all data types, including unstructured and semi-structured data such as images, videos, audio files, and documents, which are crucial for modern machine learning and advanced analytics use cases [26] [5].

Despite their benefits, many data lakes have failed to live up to their potential due to missing essential features: lack of transaction support, absence of data quality and governance enforcement, and inadequate performance optimizations. Consequently, many enterprise data lakes have turned into data swamps. Below are the descriptions of the main weaknesses of data lakes [5]:

— **Slow Performance:** as the volume of data in a data lake increases, the performance of traditional query engines often slows down. Bottlenecks include metadata management, improper data partitioning, and other issues.
— **Lack of Security Features:** data lakes are hard to properly secure and govern due to the lack of visibility and the ability to delete or update data. These limitations make it challenging to meet regulatory requirements.
— **Reliability Issues:** without the proper tools, data lakes can suffer from reliability issues that make it difficult for data scientists and analysts to work with the data. These issues can arise from difficulty in combining batch and streaming data, data corruption, and other factors.

For these reasons, a traditional data lake alone is not sufficient to meet the needs of businesses looking to innovate. Consequently, businesses often operate in complex architectures with data siloed in different storage systems: data warehouses, databases, and other storage systems across the enterprise. Simplifying this architecture by unifying all data in a data lake is the first step for companies aspiring to harness the power of machine learning and data analytics to succeed in the next decade [5].

1.3. DATA LAKEHOUSE

The solution to data lake challenges is the lakehouse, which incorporates a transactional storage layer. A lakehouse leverages data structures and management features akin to those in a data warehouse but operates directly on cloud data lakes. This integration enables traditional analytics, data science, and machine learning to function within the same system using an open format [5].

Data lakehouse is a term recently introduced in the field of large-scale data processing and storage. It is a data management system that utilizes low-cost direct access storage along with typical Database Management System (DBMS) features like Atomicity, Consistency, Isolation, and Durability (ACID) transactions, data version management, auditing, indexing, caching, and query optimization. Data lakehouses are particularly well-suited for cloud environments with separate computation and storage capabilities, allowing different compute applications to run on independent compute nodes as needed. A data lakehouse serves as a central repository for managing and storing vast amounts of both unstructured and organized data. It combines the scalability and flexibility of data lake systems with the structuring and organizing capabilities of data warehouses. This system retains data in its raw form as well as in transformed, cleaned, and structured formats, facilitating easier analysis and decision-making. Consequently, the lakehouse stands as an alternative to both systems (data warehouse and data lake), as depicted in Fig. 1.3 [8].

Currently, many businesses use multiple data warehouses in conjunction with a large data lake, eliminating redundant data and improving data quality. By linking the data lake with an ETL tool, a pipeline is established between the raw lake layer and the integrated warehouse layer. The heterogeneous data in the data lake can follow two routes: real-time processing by advanced tools like machine learning and data science, or import into the ETL, processing, and transformation into analytical data [8].

The advantages of using a data lakehouse are numerous. For data management, it provides a single source of truth, whereas raw data in a traditional data lake can cause data silos and inconsistencies. Traditional data warehouses control and structure data but become difficult to scale and manage as data volumes grow. A lakehouse, with its unified data management platform, eliminates data silos and ensures data consistency. In terms of performance, the data lakehouse offers rapid and efficient data extraction, supported by the high quality of recently transformed, purified, and organized data [8].

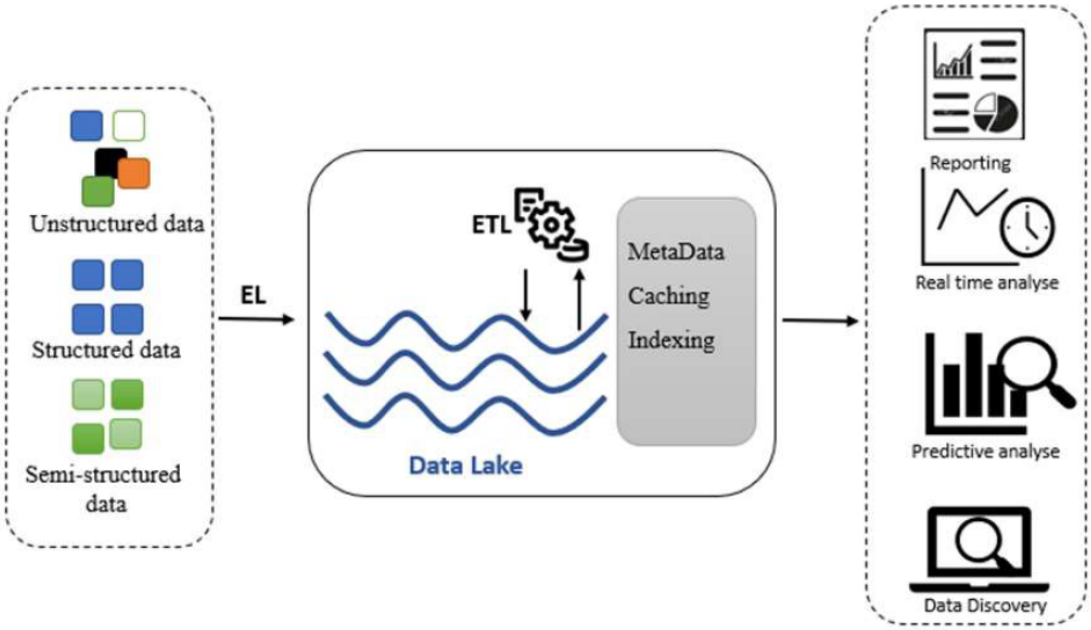


Fig. 1.3: Data lakehouse architecture [8]

The following table 1.1 shows a comparison of data lakehouses, data warehouses, and data lakes.

Table 1.1: Comparison of Data Lake, Data Lakehouse, and Data Warehouse [5]

	Data Lake	**Data Lakehouse**	**Data Warehouse**
Types of Data	All types: Structured data, semi-structured data, unstructured (raw) data	All types: Structured data, semi-structured data, unstructured (raw) data	Structured data only
Cost	Low	Low	High
Format	Open format	Open format	Closed, proprietary format
Scalability	Scales to hold any amount of data at low cost, regardless of type	Scales to hold any amount of data at low cost, regardless of type	Scaling up becomes exponentially more expensive due to vendor costs
Intended Users	Limited: Data scientists	Unified: Data analysts, data scientists, machine learning engineers	Limited: Data analysts
Reliability	Low quality, data swamp	High quality, reliable data	High quality, reliable data
Ease of Use	Difficult: Exploring large amounts of raw data can be difficult without tools to organize and catalog the data	Simple: Provides simplicity and structure of a data warehouse with the broader use cases of a data lake	Simple: Structure of a data warehouse enables users to quickly and easily access data for reporting and analytics
Performance	Poor	High	High

1.4. DATA CLOUD PLATFORMS

A cloud data platform is designed to store and manage large volumes of business data from multiple sources, allowing users to access and utilize data stored in the cloud. These sources can include databases, big data applications, and other connected devices. This flexible solution enables companies to quickly and easily process and derive insights from their digital information. Cloud data platforms are rapidly replacing traditional, on-premises warehouses. While they retain some basic functionalities, there are numerous advantages to choosing a cloud data platform. For organizations aiming to thrive in the digital age, a

cloud data platform is a crucial tool. With the appropriate platform, businesses can ensure their data remains secure, accessible, and reliable [13].

Below are the key features of data cloud platforms [13]:

— **Scalable Storage:** cloud data platforms allow businesses to scale storage according to their needs. This negates the necessity for organizations to invest in and maintain extensive hardware for data storage, leading to long-term cost savings.
— **Automated Backup:** cloud data platforms offer peace of mind by ensuring data safety and security. Regular automated backups minimize the risk of data loss due to human error or unexpected events.
— **Accessibility:** these platforms enable easy data access from anywhere at any time. Organizations can share important information with employees, customers, and partners without the need for long-distance travel or complex file-sharing methods.
— **Security:** multiple security layers in cloud data platforms protect valuable data from malicious threats. Features such as encryption, authentication, and authorization ensure the confidentiality of sensitive information.
— **Cost Savings:** cloud data platforms provide significant cost savings by eliminating the need for expensive hardware and reducing IT resource expenditures. Businesses benefit from cloud-based services without the necessity of in-house personnel.
— **Reliability:** designed for high reliability and uptime, cloud data platforms allow organizations to access critical information without concerns about outages or system failures. This continuity ensures smooth operations, enhancing productivity and reducing costs.

The top data cloud platform vendors are Snowflake, Databricks, Google BigQuery, Amazon Redshift, and Azure Synapse. In the scope of this thesis, there will be a comparison between Snowflake and Databricks [13].

1.4.1. Snowflake Data Platform

Snowflake launched in 2014 as a cloud-native, elastic data warehouse specifically designed for the cloud. Its architecture is unique because it separates storage from computing. Snowflake uses cloud object storage services like Amazon S3 as its primary data repository, and its storage layer operates independently from its SQL query processing engine, which runs on clusters of compute resources. This design provides great flexibility, allowing storage and computing to scale separately. Workloads can utilize virtual warehouse clusters that scale elastically to meet demand. When additional processing power is no longer needed, these warehouses can be dynamically suspended or resized to optimize costs. Snowflake also features zero-copy cloning and time travel on historical data without duplicating storage [7] [1].

Snowflake operates on a Software as a Service (SaaS) business model, that means it operates fully on cloud infrastructure, with all components of its service, aside from optional

command line clients, drivers, and connectors, running in public cloud environments. It relies on virtual compute instances for processing and a storage service for persistent data storage. Snowflake cannot be deployed on private cloud infrastructures, whether on-premises or hosted. Moreover, Snowflake is not a software package that users can install; it manages all software installation and updates autonomously [27] [1].

The architecture for Snowflake is designed to be enterpriese-ready service. To achieve this, Snowflake uses service-oriented architecture comprising highly-tolerant and independently scalable componentes, that communicate via RESTful interfaces. These components are organized into three layers [4]:

— Data Storage: This layer stores table data and query results.
— Virtual Warehouses: This layer, which act as a „muscle" of a system, coordinates query execution among elastic groups of virtual machines referred to as virtual warehouses.
— Cloud Services: This layer, which act as a „brain" of a system, manages virtual warehouses, queries, transactions and the metadata.

The figure 1.4 represents architecture of Snowflake with three architectural layers and their principal components:

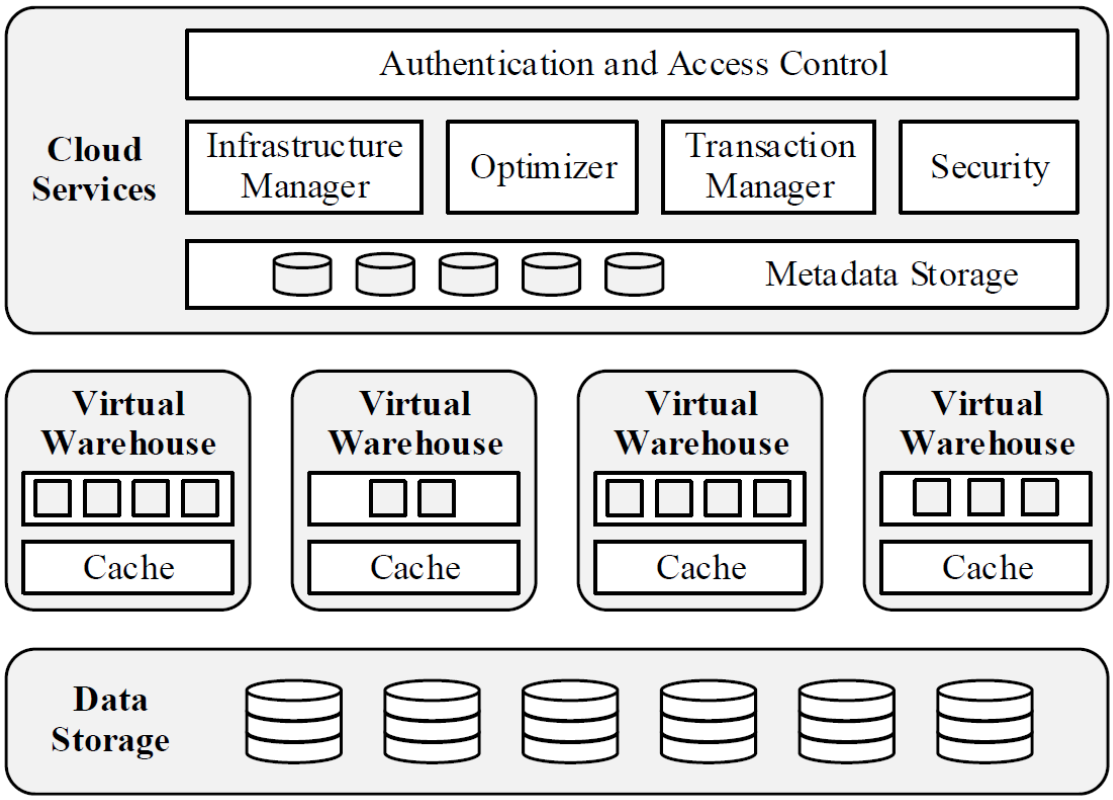


Fig. 1.4: Snowflake architecture [4]

1.4.2. Databricks Data Platform

Databricks is a unified analytics platform powered by Spark, designed to simplify the analytic processes for its customers. It utilizes its own version of Spark, known as Databricks Runtime (DBR), which introduces new features and improved performance compared to the open-source version. A significant enhancement is that it enables Spark to operate in a cloud environment, where computations occur on clusters that are started on-demand and are separate from the storage layer. DBR also incorporates cloud-specific optimizations, such as autoscaling and a Parquet cache (DBIO) on local temporary compute storage. Operating in the cloud allows Databricks to more easily monetize each customer's usage. While certain services run on the Databricks account, the primary compute and storage resources are utilized on customer accounts [20] [29].

Apache Spark is an advanced analytics engine tailored for large-scale data processing. It provides high-level APIs in Java, Scala, Python, and R, and features an optimized engine that supports general execution graphs. Spark comes equipped with a powerful set of higher-level tools: Spark SQL for SQL and structured data processing, pandas API on Spark for handling pandas workloads, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for incremental computation and stream processing [2]. Databricks architecture operates using a control plane and a compute plane [5]:

— The control plane consists of the backend services managed by Databricks. This includes the web application.
— The compute plane is responsible for processing data.

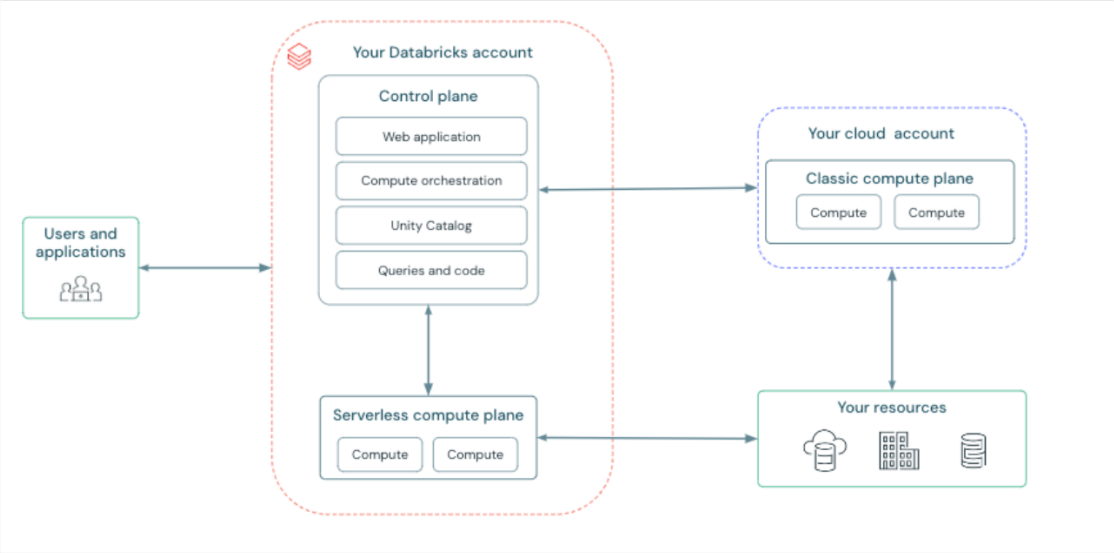


Fig. 1.5: Databricks architecture [5]

The diagram 1.5 illustrates the overall architecture of Databricks. In the serverless compute plane, Databricks resources run within a compute layer managed by Databricks in

the same AWS region as the workspace's classic compute plane. To protect customer data in this serverless environment, Databricks operates within a network boundary specific to the workspace. Multiple security layers are applied to isolate different customer workspaces, and additional network controls are in place to manage communication between clusters belonging to the same customer [5].

1.5. REFERENCE ARCHITECTURE

The comparison of Snowflake and Databricks will be carried out based on the Reference Architecture (Fig. 1.6) used by a major IT company. This architecture is used within the company to compare different tools and assist clients in choosing the best tools for specific projects. The purpose of this thesis is to compare Snowflake and Databricks using this architecture, and it will be used by the company for decision-making processes for future potential clients.

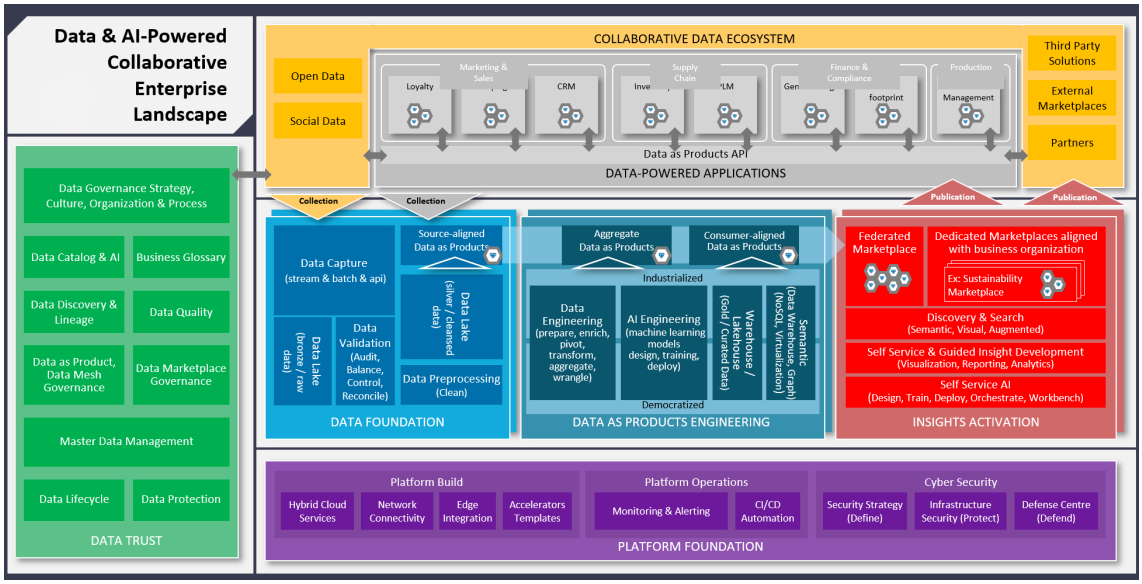


Fig. 1.6: Reference AI & Data Engineering Architecture

The scope of this thesis is to mainly focus on the most important components: data as products engineering and data foundation. The upcoming chapters will address the respective modules in the Reference Architecture and provide a detailed explanation of these aspects.

2. RELATED WORKS

This section provides an analysis of related works to find different approaches and methodologies for comparison. The analysis explores how various researchers and practitioners have handled similar problems. By reviewing these methodologies, best practices, common challenges, and potential areas for further investigation can be identified.

2.1. COMPARISON OF LAKEHOUSE STORAGE SYSTEMS

The article "Analyzing and Comparing Lakehouse Storage Systems" by Paras Jain et al. provides a comparative study of three leading open-source lakehouse systems—Delta Lake, Apache Hudi, and Apache Iceberg. This study dissects the architectural nuances and operational efficiencies of these systems, using both qualitative and quantitative methods to assess their performance and design trade-offs. The research highlights the evolution of lakehouse systems, which combine the flexible, cost-effective storage of data lakes with the robust, transactional capabilities of data warehouses [11].

Below is an analysis of the methodologies used for comparison in the article.

1. **Benchmark Development (LHBench):**
   - The researchers developed LHBench, a benchmarking tool specifically designed to evaluate lakehouse systems based on standard industry metrics and custom scenarios that highlight the unique capabilities and limitations of each system. Mainly focused on end-to-end performance, performence of data ingestion and performence of metadata access during query planning.
   - LHBench includes adaptations of the TPC-DS benchmark to meet to lakehouse-specific operations, such as transactions, indexing, and large-scale data management.
2. **Testing Environment:**
   - All tests were conducted using Apache Spark on AWS Elastic MapReduce to ensure a standardized computational environment, minimizing bias toward any specific lakehouse format.
   - This setup helps isolate performance characteristics attributable solely to the storage system designs rather than differences in underlying compute resources.
3. **Feature Comparison:**
   - An in-depth analysis of key architectural features like transaction coordination, metadata management, and data update strategies was conducted.

— The study meticulously documented how each system handles ACID transactions, metadata storage, query performance, and updates.

This article serves as a comprehensive resource for understanding the comparative strengths and weaknesses of modern lakehouse systems, providing valuable insights for organizations looking to optimize their big data infrastructure.

2.2. ASSESSING THE LAKEHOUSE

The article "Assessing the Lakehouse: Analysis, Requirements and Definition" by Jan Schneider et al. explores the concept of lakehouses —integrated data platforms that aim to combine the beneficial characteristics of both data warehouses and data lakes into a singular architecture. This paper addresses the need for a clearer definition of lakehouses, as the term is often used inconsistently across different contexts, sometimes influenced by marketing rather than technical accuracy. The authors provide a comprehensive review of the existing literature and propose a refined definition based on technical requirements that a true lakehouse should meet. They also apply these requirements to evaluate popular data management tools like Delta Lake, Snowflake, and Dremio to determine if they can genuinely support the construction of a full-fledged lakehouse [22].

The authors of the article employed a systematic and multi-faceted methodology to develop a clearer understanding of lakehouse architectures and to assess various data management tools against the lakehouse criteria. Below is an analysis of the methodology used in their study:

1. **Initial Research and Review of Existing Definitions:**
   — The first step involved an extensive review of existing literature on lakehouses, including academic papers, industry reports, and technical documentation of various data management systems.
   — This phase included parsing through publications from well-known databases, industry whitepapers, and contributions from prominent data architecture conferences.
   — The aim was to gather all relevant definitions and conceptual understandings of lakehouses, noting inconsistencies and marketing-driven definitions that diverged from technical realities.
2. **Synthesis of a Comprehensive Definition:**
   — Using insights gained from the literature review, the authors synthesized a refined definition of lakehouses in chapter 4.
   — This new definition was designed to be technologically rigorous and inclusive of essential characteristics that distinguish lakehouses from traditional data warehouses and data lakes.

— The definition emphasized the integration capabilities of lakehouses in supporting diverse analytical workloads on a unified platform, blending the flexibility of data lakes with the structured querying capabilities of data warehouses.

3. **Evaluation Framework for Data Management Tools:**
   — An evaluation framework was constructed to apply the defined technical requirements to popular data management tools like Delta Lake, Snowflake, and Dremio.
   — This framework detailed how each requirement would be tested and what features or capabilities of the tools would indicate compliance.
   — The assessment involved practical testing where possible, supplemented by a thorough review of technical documentation and whitepapers provided by the tool vendors to understand the underlying architectures and operational capabilities.
4. **Analysis and Synthesis of Evaluation Results:**
   — Results from the evaluations were meticulously compiled and analyzed, focusing on identifying which tools met the lakehouse criteria, described in chapter 4 and which fell short.
   — The synthesis of results included a comparative analysis that highlighted the strengths and weaknesses of each tool in context to lakehouse capabilities.
   — This analysis provided a comprehensive understanding of the current landscape of data management solutions and their alignment with the lakehouse architecture.

This detailed methodology not only provided a robust basis for defining and understanding lakehouses but also offered a practical approach to evaluating current tools against these new standards, making a significant contribution to the field of data architecture.

2.3. MANAGING CLOUD-BASED BIG DATA PLATFORMS

The article "Managing Cloud-Based Big Data Platforms: A Reference Architecture and Cost Perspective" by Leonard Heilig and Stefan Voß presents a comprehensive framework for implementing big data applications using cloud services. The authors propose a generic reference architecture that integrates state-of-the-art cloud services to support big data processing phases, including data generation, ingestion, storage, analytics, and presentation. The study focuses on the implementation of this architecture using the cloud services of Amazon Web Services (AWS), Google Cloud, and Microsoft Azure [9]. Below is a description of the key methodologies used for comparison in the article:

1. **Generic Reference Architecture:**
   — The authors define a generic reference architecture to manage big data applications. This architecture supports batch and stream processing and includes phases such as data generation, ingestion, storage, analytics, and presentation.

— They reviewed technical documents, recommendations, and use cases from cloud providers and interviewed experts to develop this architecture.

2. **Cost Analysis:**
   — The study investigates the pricing schemes and cost factors associated with each cloud provider's services.
   — A case study focusing on real-time data streaming is used to compare costs across different cloud services.
   — The analysis includes different components such as data streams, stream processing, storage, Hadoop clusters, data warehouses, and machine learning services.
3. **Comparison of Key Cost Factors:**
   — Data Streams and Stream Processing: Costs are analyzed based on message events and throughput capacity, with specific comparisons of Amazon Kinesis Streams, Google Cloud Pub/Sub, and Azure Event Hubs.
   — Data Storage: Costs are compared considering storage capacity, data request types, and data transfer costs. The analysis includes Amazon S3, Google Cloud Storage, and Azure Storage.
   — Hadoop Clusters: Costs for running Hadoop clusters are compared based on VM types, storage, and additional usage fees, considering services like Amazon EMR, Google Cloud Dataproc, and Azure HDInsight.
   — Data Warehouses: The pricing approaches for data warehouse services such as Amazon Redshift, Google BigQuery, and Azure SQL Data Warehouse are compared.
   — Machine Learning Services: The cost factors for machine learning services from AWS, Google, and Azure are analyzed, focusing on computing capacity, number of transactions, and subscription fees.

2.4. APPLYING REVIEWD METHODOLOGIES IN THESIS RESEARCH

The methodology derived from the related works will be highly beneficial for comparing Snowflake and Databricks.

The initial research and review of existing definitions methodology from Section 2.2 is used in this thesis to examine existing literature and establish fundamental definitions for data warehouse, data lake, and data lakehouse - see Chapter 1.

Very important methodologies used in this thesis are the "evaluation framework for data management tools" and "feature comparison." For each aspect in the reference architecture, there is first an analysis of the available tools and features, followed by a comparison of these features. The article from Section 2.1 suggests which areas the comparison can focus on, such as query performance, ACID transactions, and so on. The article from Section 2.2 indicates that the assessment involved practical testing where possible, supplemented by a thorough review of technical documentation and whitepapers provided by the tool vendors

to understand the underlying architectures and operational capabilities. In this thesis, the feature analysis is mainly based on Databricks and Snowflake documentation.

The article from Section 2.3 can be the most useful in this thesis because it also uses reference architecture. Based on the analysis of the methodology, it will help to conduct the research based on reference architecture. It also describes cost analysis, which is also conducted in this thesis for the comparison of Cortex and AutoML - see Section 3.1.5 and Section 3.1.6.

3. DATA AS PRODUCTS ENGINEERING

The first area for comparison between Snowflake and Databricks, according to AI & Data Engineering Reference Architecture 1.6, is "Data as Products Engineering." Begin by assessing the tools available for both platforms in each aspect, then decide on the comparison criteria and methods.

3.1. AI ENGINEERING

The first aspect for analysis from "Data as Products Engineering" is AI Engineering. First, there will be an analysis of available tools in Databricks and Snowflake, followed by a comparison between them.

3.1.1. AI Engineering - Databricks

The diagram 3.1 demonstrates how components collaborate together to support the implementation of machine learining model development and deployment processes.

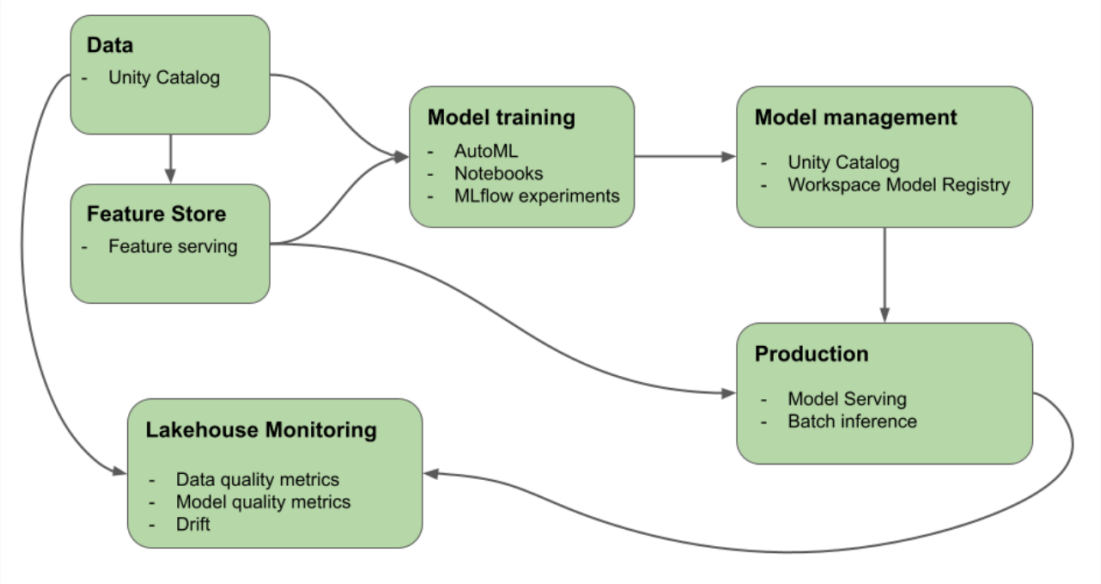


Fig. 3.1: AI and ML model development and deployment process [5]

In this thesis, the focus will primarily be on model training, model management, and deployment (production). The following tools are available for model training in Databricks: AutoML, Notebooks, and MLflow experiments. Below is the description of these tools.

AutoML facilitates the automatic application of machine learning by taking a supplied dataset and a defined prediction target to prepare for model training. It executes and logs multiple trials to develop, adjust, and assess various models. After evaluating the models, it displays the results and provides a Python notebook with the source code of each trial to enable code review, reproduction, and alterations. AutoML further generates and saves summary statistics of the dataset in a notebook for subsequent analysis. It is useful for regression, classification, and forecasting challenges [5].

Notebooks serve as the essential tool for developing data science and machine learning workflows and facilitating collaboration among colleagues. Databricks notebooks offer real-time co-authoring in various languages, automatic versioning, and integrated data visualizations [5].

Databricks Machine Learning extends the fundamental capabilities of the platform with a suite of specialized tools for data scientists and machine learining (ML) engineers, such as MLflow and Databricks Runtime for Machine Learning [5].

MLflow is an open-source platform designed to manage the entire machine learning lifecycle. Therefore, it is possible to integrate MLflow with Databricks for use in model training, management, and deployment [5].

In MLflow, **experiments** serve as the fundamental organizational unit, with every MLflow run being part of an experiment. Each experiment enables the visualization, searching, and comparison of runs, in addition to allowing the downloading of run artifacts or metadata for further analysis in other tools. These experiments are hosted on a Databricks-managed MLflow tracking server [5].

The diagram 3.2 indicates that for model management, there is the Unity Catalog and the Workspace Model Registry. However, since April 2024, Databricks has disabled the Workspace Model Registry, making the Unity Catalog the default catalog. Unity Catalog offers centralized access control, auditing, lineage, and data discovery capabilities across Databricks workspaces.

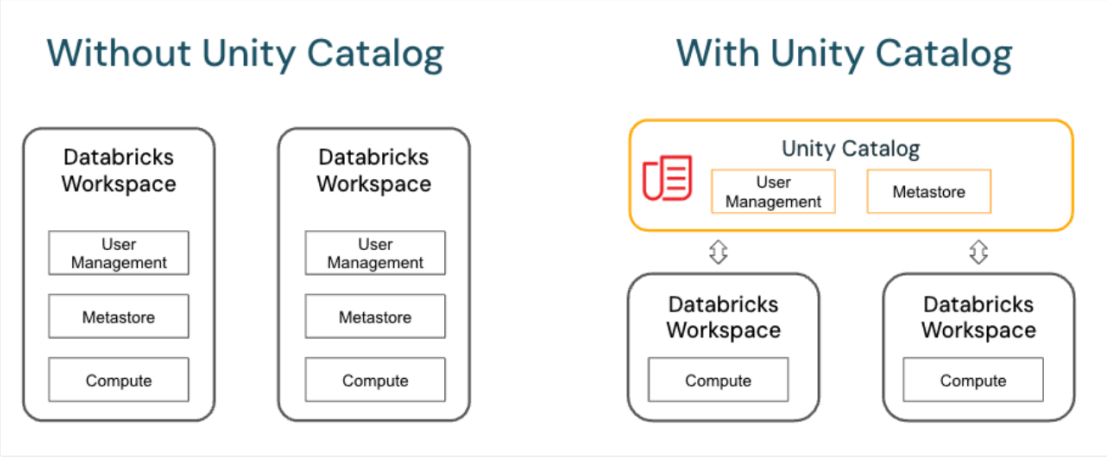


Fig. 3.2: Unity Catalog Architecture [5]

Key features of Unity Catalog include [5]:

— **Standards-compliant security model:** based on standard ANSI SQL, Unity Catalog's security model permits administrators to set permissions in their existing data lakes using well-known syntax at the catalog, database (or schema), table, and view levels.
— **System tables (Public Preview):** Unity Catalog provides straightforward access to and querying of your account's operational data, such as audit logs, billable usage, and lineage, which offers an extensive audit trail for data at a highly detailed level.
— **Define once, secure everywhere:** Unity Catalog serves as a single point to establish data access policies that are enforced across all workspaces.
— **Data discovery:** Unity Catalog enables the tagging and documentation of data assets, complemented by a search interface to assist data users in locating data.
— **Built-in auditing and lineage:** Unity Catalog automatically records user-level audit logs that track data access and captures lineage data detailing how data assets are developed and utilized across all languages.

For the deployment of machine learning models there is also the option of using Model Serving. Databricks Model Serving provides a comprehensive interface to deploy, govern, and query AI models, making each served model accessible via a REST API for integration into web or client applications. The figure 3.3 shows an example view of model serving in Databricks. The service ensures high availability and minimal latency in model deployment, dynamically scaling to accommodate demand variations, thus lowering infrastructure expenses and enhancing latency performance using serverless technology [5].

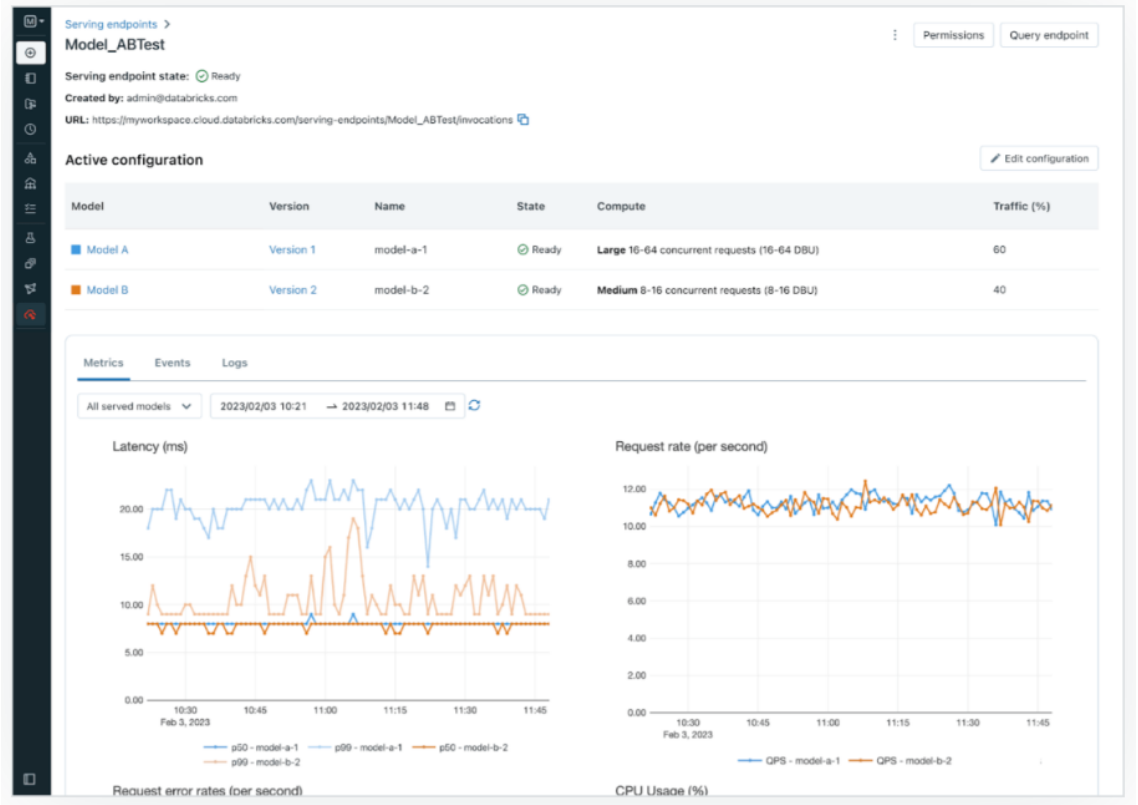


Fig. 3.3: View of Model Serving in Databricks [5]

3.1.2. AI Engineering - Snowflake

Snowflake for machine learning uses Snowpark ML, the Python library and infrastructure that facilitate end-to-end ML workflows, incorporating elements for model development and operations. Snowpark ML includes two key components: Snowpark ML Modeling (data preprocessing, model training) and Snowpark ML Operations (model management and deployment). The architecture 3.4 provides a general overview of Snowflake for Machinge Learining [27].

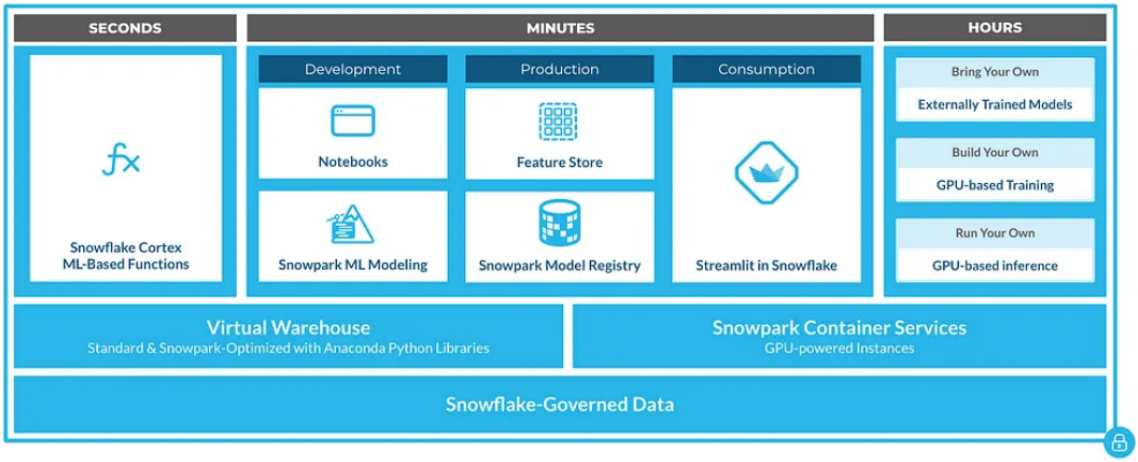


Fig. 3.4: Snowflake for Machine Learning [28]

Snowpark ML Modeling enables data preprocessing, feature engineering, and model training in Snowflake, utilizing popular machine learning frameworks such as scikit-learn, xgboost, and lightgbm. Additionally, this API includes a preprocessing module that uses compute resources from a Snowpark-optimized warehouse to perform scalable data transformations. Snowpark ML Operations (MLOps) featuring the Snowpark ML Model Registry, enhances the Snowpark ML Development API. This registry enables secure deployment and management of models in Snowflake and accommodates models trained both internally and externally. **As a component of Snowpark ML Operations** (MLOps), the Snowflake Model Registry allows for the secure management of models and their metadata within Snowflake, irrespective of their origin. The model registry classifies machine learning models as first-class schema-level objects in Snowflake, facilitating their discovery and utilization within organizations. Registries can be established and models stored using classes from the Snowpark ML library. Models may have multiple versions, with the option to designate a version as the default. Snowflake also introduced a new feature on March 12, 2024, known as Cortex, which is now available for use. Snowflake Cortex is an intelligent, fully managed service that provides machine learning and AI solutions to Snowflake users. The capabilities of Snowflake Cortex include [27]:

— **LLM Functions:** SQL and Python functions that utilize large language models (LLMs) for tasks such as understanding, querying, translating, summarizing, and generating free-form text.
— **ML Functions:** SQL functions designed for predictive analysis using machine learning to facilitate insights into structured data and enhance everyday analytics.

Snowflake Cortex delivers state-of-the-art ML and AI solutions within the Snowflake security perimeter. Therefore, workloads using Snowflake Cortex solutions can take advantage of Snowflake's existing security, scalability, and governance capabilities. Cortex ML functions offer analysts, data engineers, and business users access to powerful machine learning models that are optimized for scalability and accuracy. The algorithms are provided, requiring only user data. These features are accessible through SQL:

— Forecasting
— Anomaly Detection
— Classification

3.1.3. Plan for Conducting the Research - AI Engineering

In the previous sections, available tools and approaches for ML in Snowflake and Databricks were presented. It was concluded that these approaches are different and many aspects depend on the implementation of these tools, making direct comparison difficult. After analysis, the best way to compare ML in Databricks and Snowflake is by comparing AutoML and Cortex, as these tools automatically generate code for training the models

and saving the results. This comparison will determine which tool is better. The common features of Cortex and AutoML are classification and forecasting, so these two operations will be compared. The experiment will be conducted on generated test data in three scenarios: 1,000, 10,000, and 100,000 rows of test data. The goal is to generate simple test data and test how Databricks and Snowflake can handle it. Using information from the analysis in related works, the experiment will primarily compare performance, costs, and corresponding indicators of efficiency for models. For classification, the accuracy will be measured, and for forecasting, several different metrics will be used to evaluate performance. For Databricks, it was necessary to create a cluster. For this experiment, a cluster was created with the following configuration:

— Node Type: Standard_D4s_v3 (16GB RAM, 4 cores)
— Runtime Version: 15.3 ML Beta (Apache Spark 3.5.0, Scala 2.12) - this type of runtime version supports Unity Catalog
— Resource Class: single node

For Snowflake, it was only necessary to choose the size of the warehouse. For this experiment, the XS size was chosen, which has 16GB RAM and 8 cores. For cost comparison, it is necessary to first describe how pricing works in Databricks and Snowflake.

3.1.4. Pricing in Snowflake and Databricks

Snowflake's platform powers the Data Cloud by simplifying operations and reducing costs, addressing the limitations of other solutions. As a service available on leading cloud providers, Snowflake eliminates architectural complexity, enabling enterprises to run diverse workloads with the necessary elasticity, performance, and scale. Snowflake's architecture divides data warehousing into three layers: storage, virtual warehouses (compute), and cloud services. Due to the very small sizes of the files in this thesis, the cost estimation will be based on compute. Pricing is based on the actual usage of these layers and serverless features, measured in Snowflake credits. A Snowflake credit is consumed only when resources are actively used, such as when a virtual warehouse is running, the cloud services layer is working, or serverless features are utilized. Different sizes of virtual warehouses consume Snowflake credits at specified rates (Fig. 3.5), billed per second with a minimum charge of one minute. In this thesis, a chosen XS size warehouse will consume 1 credit, which costs 3.90$ per credit for the enterprise service level in the EU Frankfurt region [27].

	XS	S	M	L	XL	2XL	3XL	4XL	5XL*	6XL*
Standard	1	2	4	8	16	32	64	128	256	512
Snowpark-Optimized	N/A	N/A	6	12	24	48	96	192	384	786

Fig. 3.5: Virtual Warehouses Credits per Hour [27]

In Databricks SQL, there is also a Pay-as-You-Go approach similar to Snowflake. However, for the AutoML experiments, it was necessary to run them in a cluster where the payment is calculated based on the time the cluster is running. For the configuration described in the previous section, a cluster with Standard_DS3_v2 node type in the West Europe region costs $0.413 per hour.

The next subsections will present the results of these comparisons.

3.1.5. Comparison of Cortex and AutoML - Forecasting

To compare forecasting in AutoML and Cortex tools, generated test data contains only two columns: date and value. A Python script using the Pandas and NumPy libraries was used to generate this data. To make the data more realistic, a sine function pattern was simulated, with added noise and trend. The figures below show the plots of the 1,000 (Fig. 3.6), 10,000 (Fig. 3.7) and 100,000 (Fig. 3.8) generated data for forecasting. All of the generated data is available in the public repository [21].

This generated data was saved to CSV files and uploaded to the tables in the Snowflake database and Databricks Unity Catalog. The forecasting will be done in the following scenarios:

— 1,000 Rows of Data: 30 days of prediction
— 10,000 Rows of Data: 300 days of prediction
— 100,000 Rows of Data: 300 days of prediction

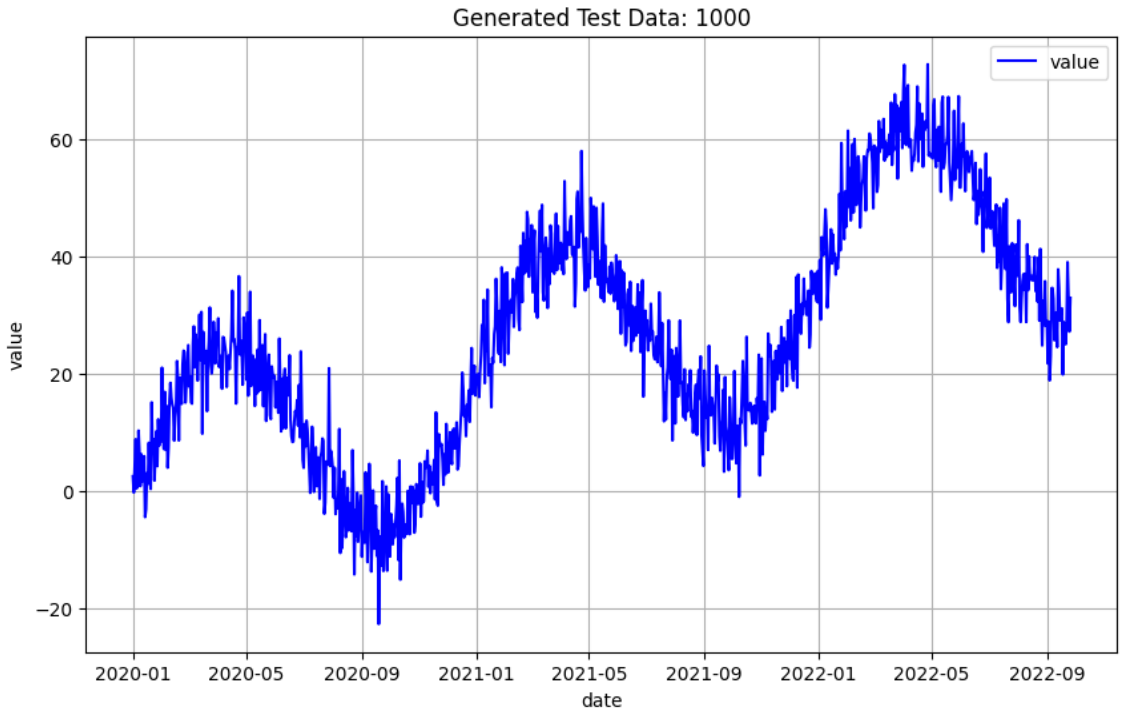


Fig. 3.6: Generated 1,000 Test Data for Forecasting

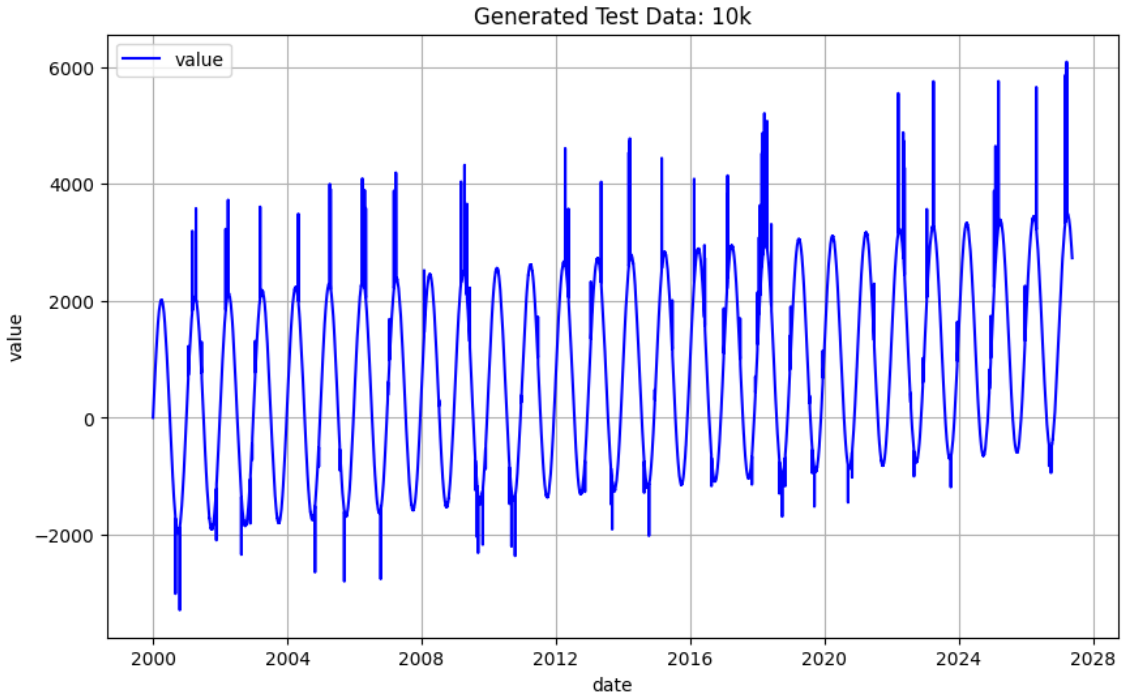


Fig. 3.7: Generated 10,000 Test Data for Forecasting

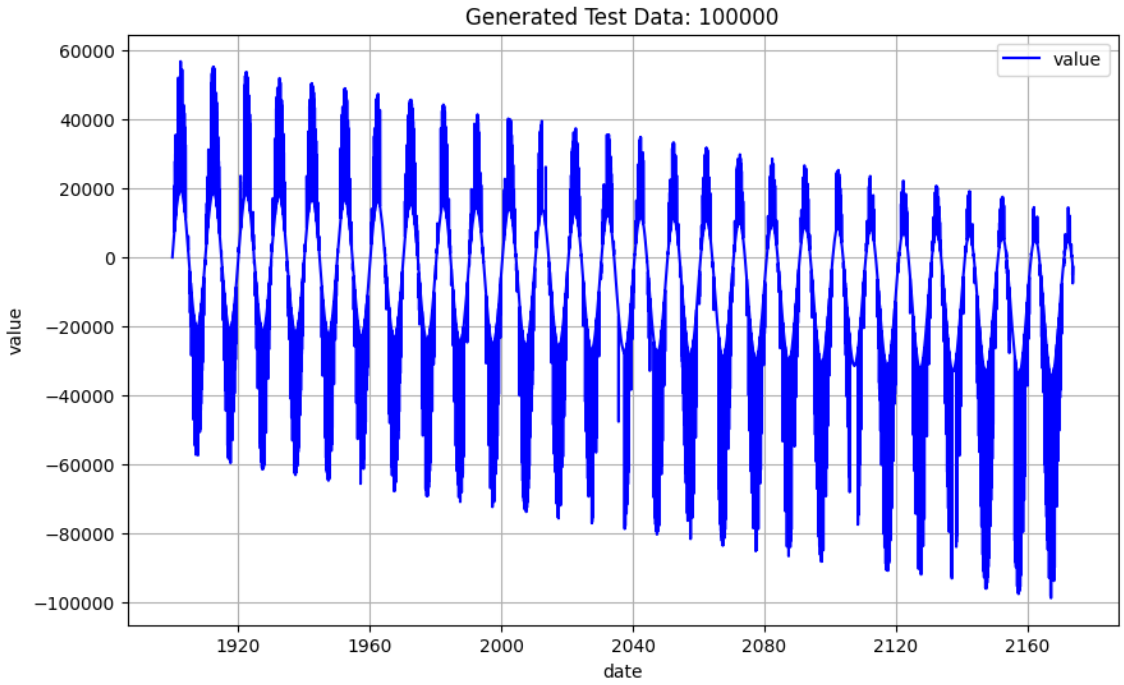


Fig. 3.8: Generated 100,000 Test Data for Forecasting

In Cortex, after choosing the prediction target and time column, there is an option to choose the forecast horizon and prediction interval width. Snowflake then automatically creates code for training the model and saving the results (Fig. 3.9). It also generates two commands to inspect the results, which show evaluation metrics and feature importance.

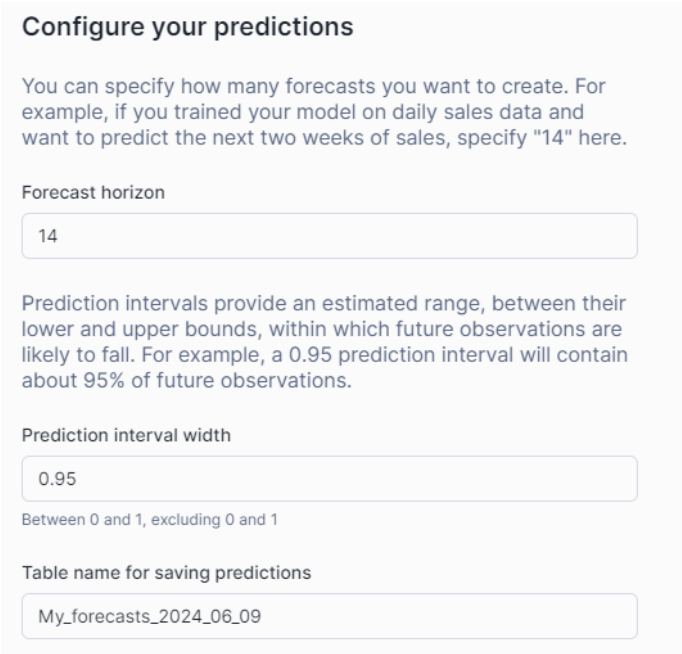


Fig. 3.9: Cortex Configuration for Forecasting

In the AutoML experiment, there was no option to choose the prediction interval width.

However, there are options to choose evaluation metrics, training frameworks, country holidays, and timeout (Fig. 3.10).

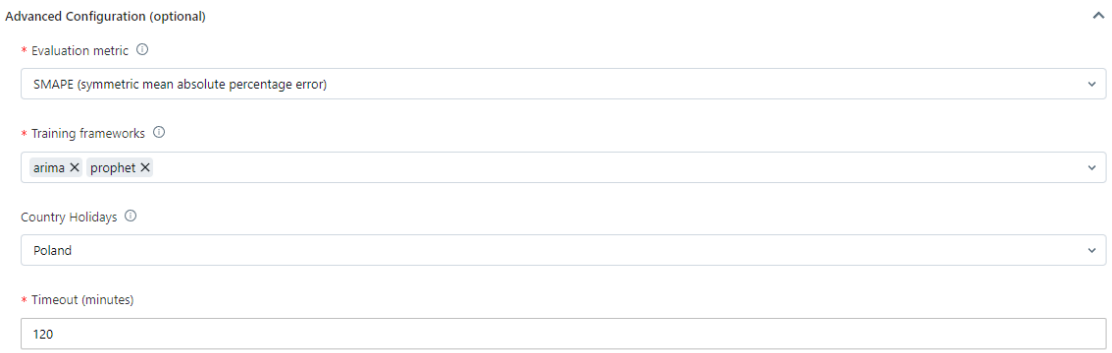


Fig. 3.10: AutoML Configuration for Forecasting

To achieve the best result from Databricks training multiple models, the configuration selects the model with the best error metric, which in this case is SMAPE (Fig. 3.11).

							Metrics	Tags
	Run Name	Created	Dataset	Duration	Source	Models	val_smape	model_type
	Prophet	4 days ago	-	3.7min	Notebo...	pyfunc	0.09333991...	prophet
	Prophet	4 days ago	-	3.5min	Notebo...	pyfunc	0.09351141...	prophet
	Prophet	4 days ago	-	4.3min	Notebo...	pyfunc	0.09353949...	prophet
	Prophet	4 days ago	-	4.0min	Notebo...	pyfunc	0.09415653...	prophet
	Arima	4 days ago	-	1.9min	Notebo...	pyfunc	0.13213863...	arima
	Training Data Storage and...	4 days ago	-	2.8s	Notebo...	-	-	-

Fig. 3.11: AutoML Experiment Runs for 1,000 Data - Forecasting

Additionally, the results include generated plots of data exploration and results, thanks to execution in a notebook sucha as forecast components and forecast with change points and trend for each scenario. The figures (Fig. 3.12, Fig. 3.13, Fig. 3.14) are automatically generated and show data points with the predicted values, where the red line represents the trend, the black points represent the data, the blue line represents the predicted values, and the light-blue area represents the uncertainty.

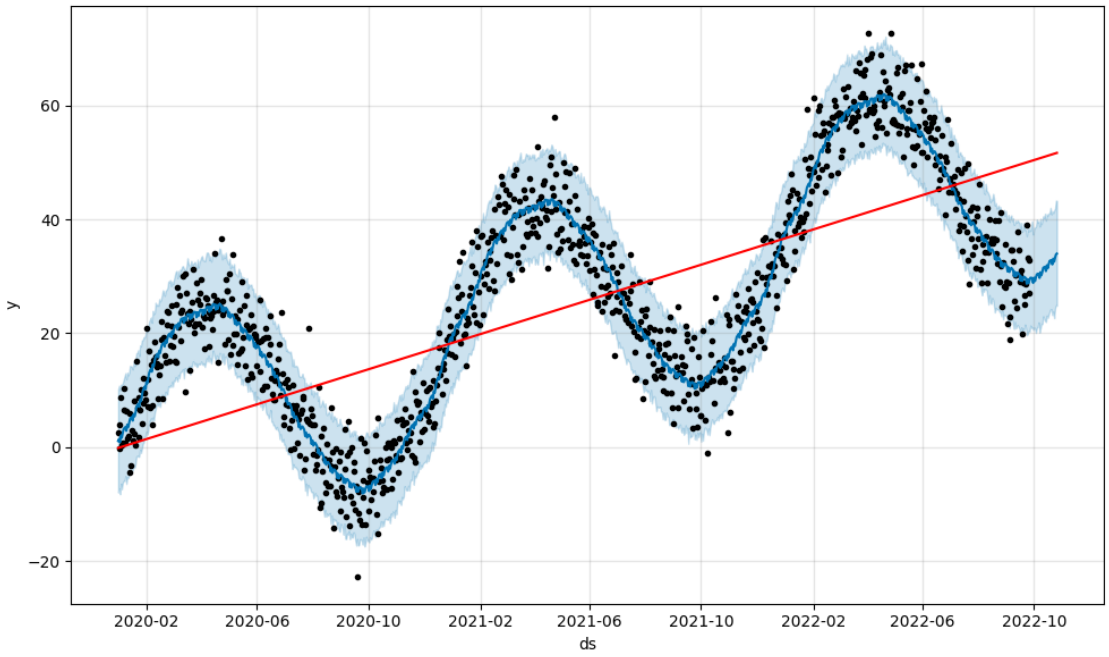


Fig. 3.12: Forecast with Change Points and Trend for 1,000 data

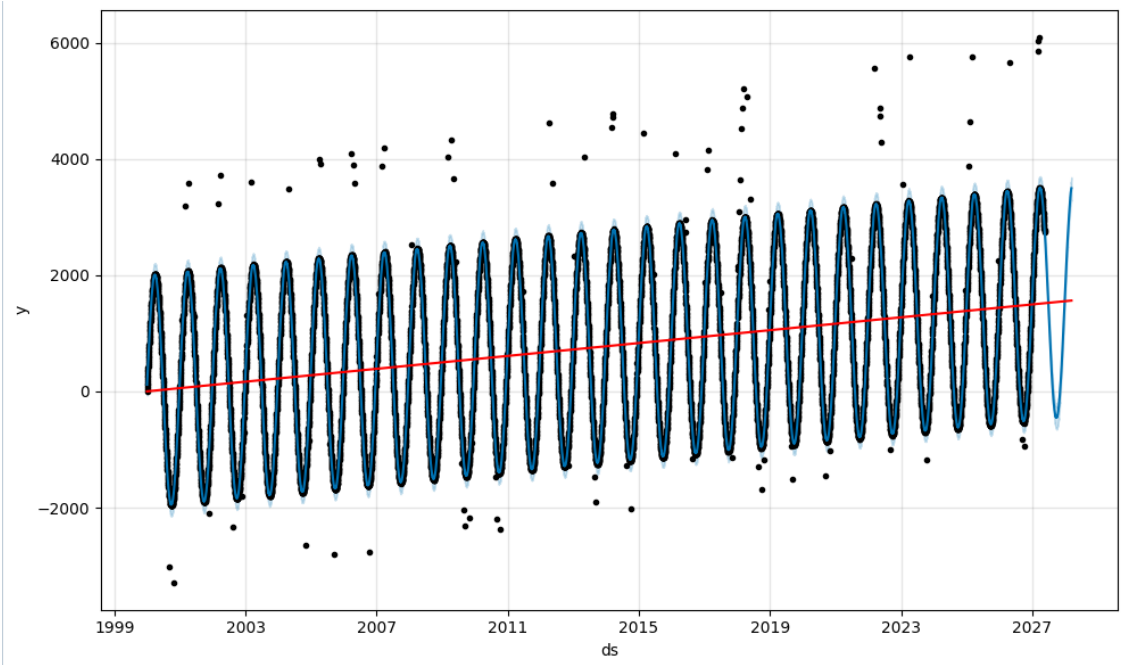


Fig. 3.13: Forecast with Change Points and Trend for 10,000 data

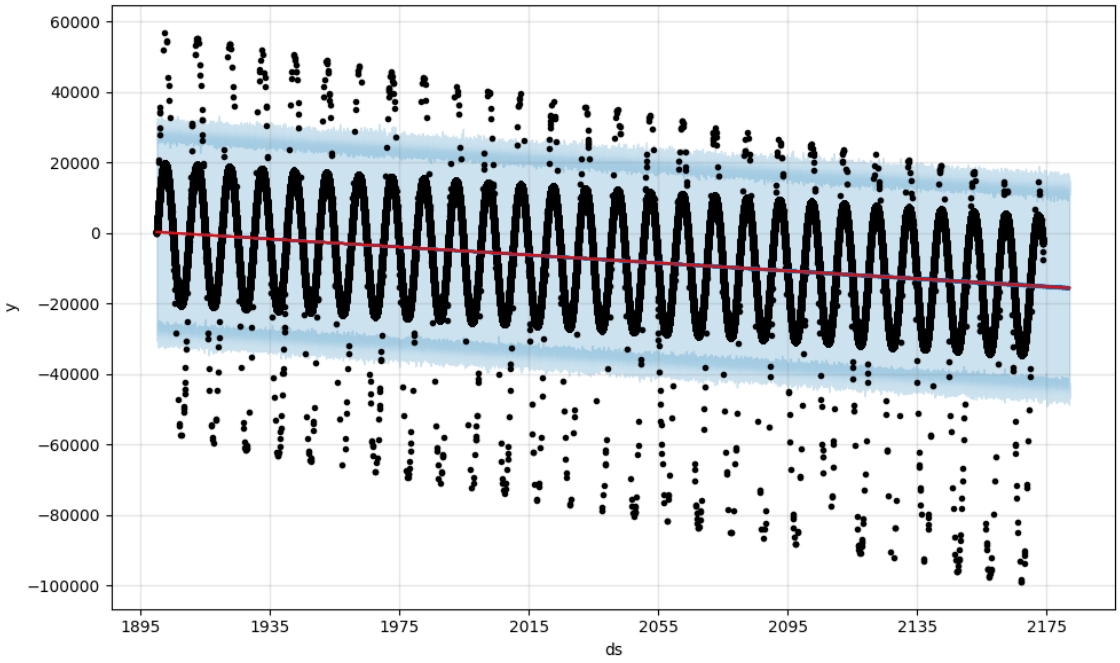


Fig. 3.14: Forecast with Change Points and Trend for 100,000 data

Visualization is also possible in Snowflake, but it requires an additional step to configure. One method to visualize results is to use Streamlit. Streamlit is a Python library that simplifies the creation and sharing of custom web apps for machine learning and data science. With Streamlit, you can swiftly develop and deploy robust data applications. For additional details about this open-source library, refer to the Streamlit Library documentation [27].

Snowflake and Databricks provide error metrics to evaluate the results of the training model and forecasting values for each scenario. Here is a description of some of them [10] [12]:

— Mean Absolute Error (MAE): This metric quantifies the average inaccuracy in predictions as a percentage, indicating the average deviation or error percentage that can be expected from the predictions.

— Mean Absolute Percentage Error (MAPE): This metric provides a better understanding of evaluation by considering the percentages of absolute error terms relative to actual values. By doing so, it eliminates the need to consider the scale of the actual values. MAPE, expressed as a percentage, allows for the calculation of prediction accuracy by subtracting it from 100. It serves as a simpler version of the Mean Absolute Error (MAE).

— Mean Square Error (MSE): This metric shares the same underlying principle as MAE, which is to avoid the cancellation of positive and negative error terms when summed together. MSE achieves this by considering the squares of the error terms and then computing their mean. It is commonly used as an error function in regression settings, which is then optimized to obtain the optimum model parameters.

— Symmetric Mean Absolute Percentage Error (sMAPE): This metric normalizes the relative errors by dividing by both the actual and predicted values, ensuring that the metric ranges between 0% and 100%.

Below is the table 3.1 with a comparison of the results in Databricks and Snowflake based on error metrics, where green color indicates a better result. For the 1,000 test data, the results are very similar, but Databricks has slightly better error metrics. For 10,000 data , the difference is larger, and Databricks also has better results. However, for these two scenarios, both data platforms perform well in forecasting. In the last scenario, the difference is the greatest, and here Snowflake works much better and has better results. The forecasting didn't work correctly in Databricks for that large amount of data, and it didn't detect the pattern as it did for the 1,000 and 10,000 test data points. The difference is visible in the figure 3.14 where the trend and prediction lines are almost the same.

Table 3.1: Comparison of the Result for Forecasting in Cortex and AutoML - Error Metrics

		Snowflake	Databricks	Snowflake	Databricks	Snowflake	Databricks
Rows		1000		10000		100000	
Error Metric	MAE	4,299	3,927	117,535	80,194	5063,354	13158,048
	MAPE	0,102	0,096	0,590	0,267	3,875	26,276
	MSE	27,620	24,745	53239,013	28601,077	60037874,522	225100700,000
	SMAPE	0,097	0,093	0,204	0,155	0,612	1,062

The table 3.2 shows the comparison of Snowflake and Databricks based on the duration of execution of the experiments and costs. Considering the duration, Snowflake performs much better in every scenario. Similarly, for cost, Databricks is cheaper only for 1,000 test data points, but the cost difference is minimal. The most significant difference is seen with 100,000 test data points, where Snowflake takes 5 minutes while Databricks takes 10 hours. This extended duration is due to a 10-hour timeout. Training the Arima framework would take even longer, but the timeout stopped the experiments and took the other best-trained model. Without Arima, the experiments would last 5.2 hours. The experiments were also conducted based on other evaluation metrics besides SMAPE, and the results were very similar.

Table 3.2: Comparison of the Result for Forecasting in Cortex and AutoML - Duration and Costs

	Snowflake	Databricks	Snowflake	Databricks	Snowflake	Databricks
Rows	1000		10000		100000	
Cost	0,106$	0,057$	0,121$	0,316$	0,324$	4,14$
Duration	1m 38s	8m 14s	1m 52s	45m 54s	4m 59s	10h 1m 15s

In conclusion, the comparative analysis of forecasting performance and efficiency between Databricks and Snowflake reveals distinct strengths and weaknesses depending on

the dataset size. For smaller datasets of 1,000 and 10,000 rows, both platforms perform well, with Databricks exhibiting slightly better error metrics and minimal cost differences. However, as the dataset size increases to 100,000 rows, Snowflake demonstrates a significant advantage in both forecasting accuracy and execution efficiency. Databricks struggles with large datasets, failing to detect patterns effectively and incurring substantially longer execution times due to a 10-hour timeout limitation. Conversely, Snowflake not only delivers superior forecasting results but also completes the task in a fraction of the time, highlighting its scalability and cost-efficiency. The findings suggest that while Databricks may be suitable for smaller-scale tasks, Snowflake is the preferable choice for large-scale data forecasting, ensuring better performance and reduced operational costs. Databricks struggle with the large datasets. The platform cannot find the patterns efficiently and time out after 10 hours. Snowflake is not just giving better forecasting, but also completing the tasks quicker. This shows how scalable and cost-effective Snowflake is.

3.1.6. Comparison of Cortex and AutoML - Classification

To compare forecasting in AutoML and Cortex tools, the generated test data contains two feature columns and one target column. A Python script using the `make_classification` function from the sklearn library was used to generate this data. The figures below show the plots of the 1,000 (Fig. 3.15), 10,000 (Fig. 3.16) and 100,000 (Fig. 3.17) generated data for classification.

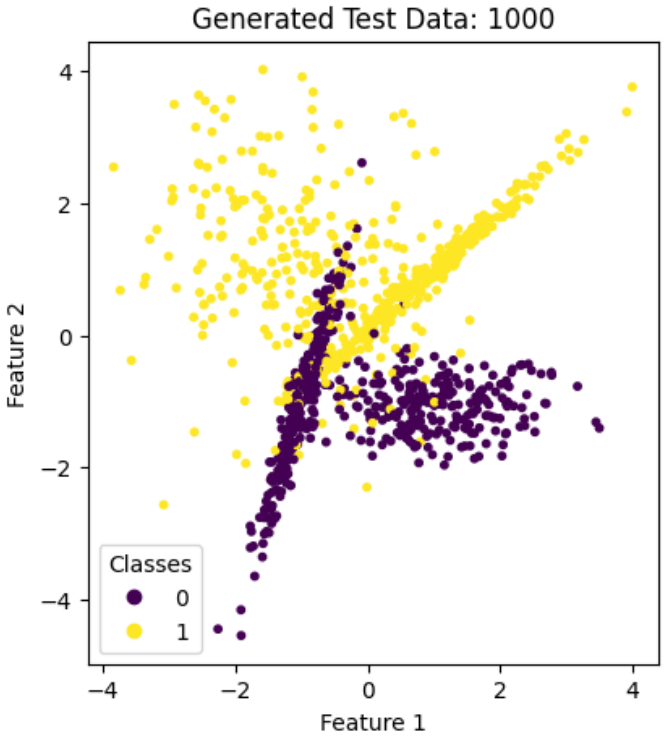


Fig. 3.15: Generated 1,000 Test Data for Classification

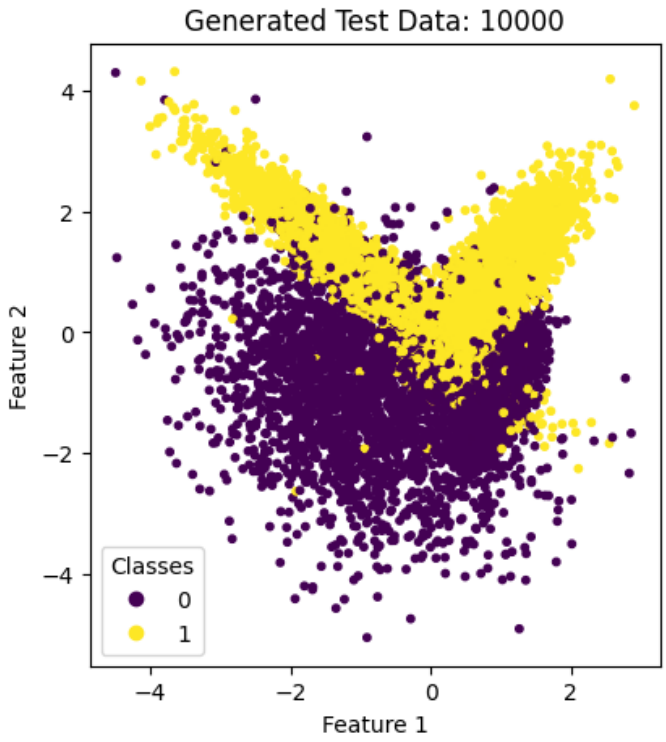


Fig. 3.16: Generated 10,000 Test Data for Classification

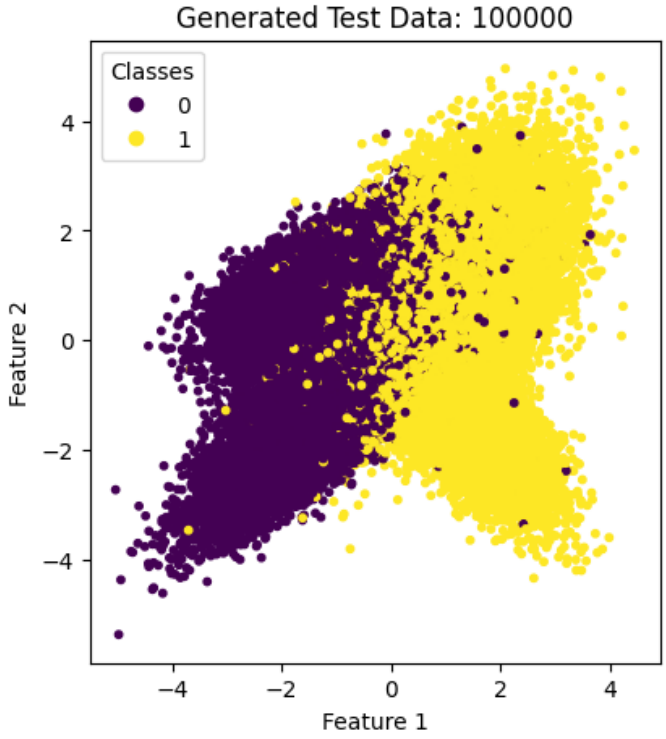


Fig. 3.17: Generated 100,000 Test Data for Classification

In Cortex, it was only necessary to provide the input training dataset and select the prediction target. The rest of the operations were done automatically by Cortex, and it generated commands to inspect the following results: evaluation metrics, global evaluation metrics, confusion matrix (Fig. 3.18), and feature importance.

In AutoML, there is also an additional option to choose the evaluation metric and training frameworks for forecasting. For this experiment, the F1 score was chosen as the evaluation metric, and LightGBM, sklearn, and XGBoost were chosen as the training frameworks. Similar to forecasting, for classification, AutoML also generated some data visualizations such as a confusion matrix (Fig. 3.19), ROC curve, and precision-recall curve.

	DATASET_TYPE	ACTUAL_CLASS	PREDICTED_CLASS	COUNT
1	EVAL	0	0	93
2	EVAL	0	1	8
3	EVAL	1	0	8
4	EVAL	1	1	91

Fig. 3.18: Confusion Matrix Generated in Cortex for 1,000 data

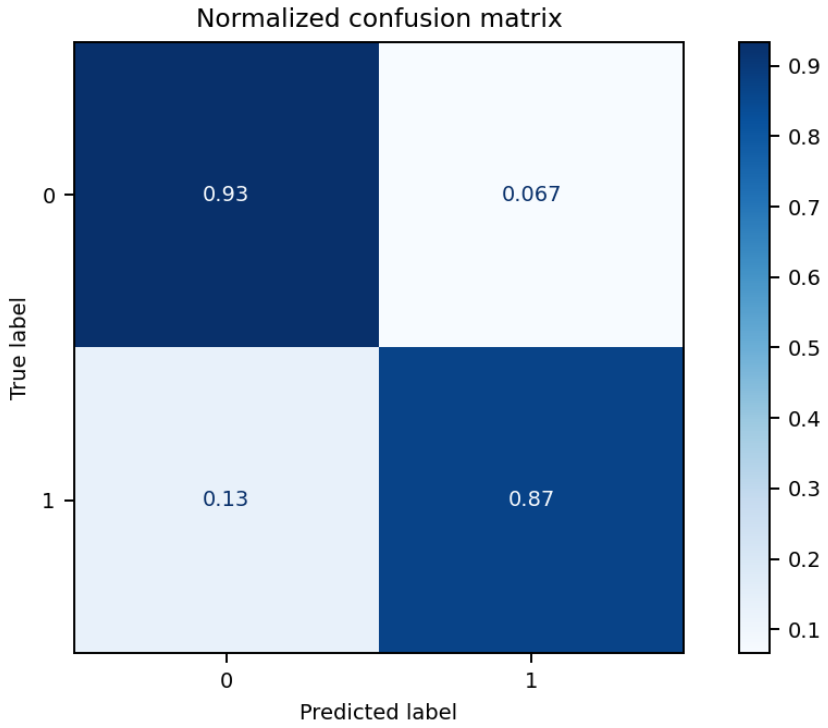


Fig. 3.19: Confusion Matrix Generated in AutoML for 1,000 data

Snowflake and Databricks provide error metrics to evaluate the results of the training model and forecasting values for each scenario. Here is a description of some of them [16] [17]:

— Accuracy: This is the most common performance metric used for checking a binary classification model. It show ratio of correct predictions to total predictions. High

acuracy means the model predicting large proportion of instances correctly, while low acuracy means the model makes too many incorrect predictions.

— F1 score: This is a performance metric that integrates both precision and recall to offer a comprehensive assessment of a binary classification model's performance. It calculates the harmonic mean of precision and recall, thereby assigning equal significance to both metrics.

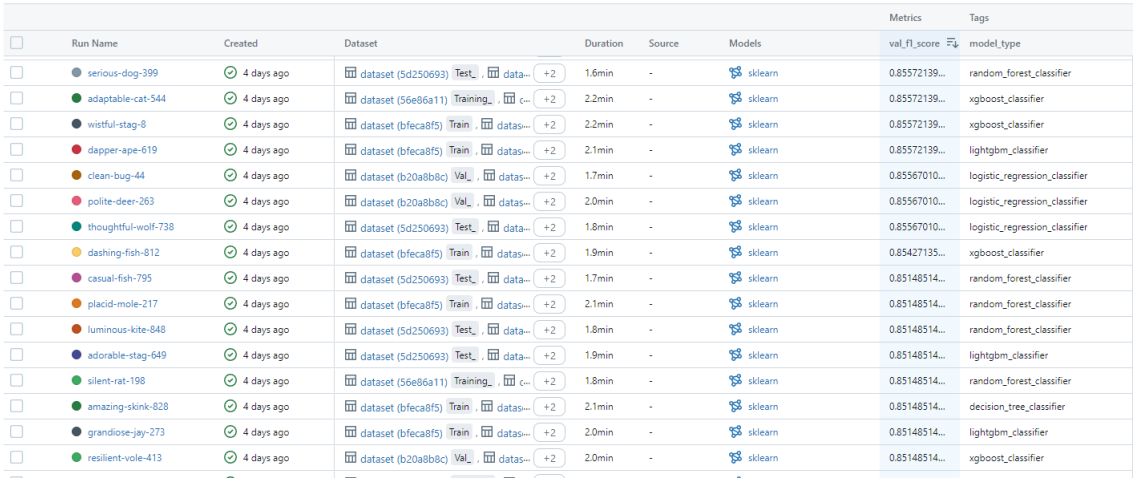
— Log_loss: This is a log transformation of the likelihood function, often used to evaluate probabilistic classifiers. Unlike accuracy and similar metrics, Log Loss accounts for the uncertainty in predictions, penalizing models more harshly for confidently incorrect predictions.

The table 3.3 similar to the forecasting table, represents the results from Cortex and AutoML based on the error metrics described above. The classification in Cortex works better for 1,000 and 10,000 data points. Snowflake has slightly better error metrics, but both Cortex and AutoML produce correct results. The results differ for the 100,000 data points, where AutoML has much better results than Cortex. In Snowflake, for that large dataset, the classification results failed and only achieved around 58% accuracy.

Table 3.3: Comparison of the Result for Classification in Cortex and AutoML - Error Metrics

		Snowflake	Databricks	Snowflake	Databricks	Snowflake	Databricks
Rows		1000		10000		100000	
Error Metric	Accuracy	0,920	0,903	0,939	0,931	0,578	0,873
	F1 score	0,920	0,910	0,940	0,931	0,578	0,877
	Log_loss	0,217	0,314	0,173	0,210	0,670	0,296

In the table 3.4 there is a comparison between Cortex and AutoML based on duration and costs. Snowflake has better results in each scenario, and the costs and duration are very similar regardless of the size of the datasets. The same is true for Databricks, where for each scenario, it takes around 2 hours. This is due to a timeout set to 2 hours. Databricks trains many different models and chooses the best one, similar to the process in forecasting. It includes many model types, such as XGBoost, LightGBM, logistic regression, random forest, and decision tree. This is presented in the figure 3.20. However, for classification, it trains many more models, and after 2 hours, it selects the best one. Therefore, it is important to correctly configure the timeout. Below in the figure, there is a view of some experiments run for classification.



	Run Name	Created	Dataset	Duration	Source	Models	Metrics: val_f1_score	Tags: model_type
	serious-dog-399	4 days ago	dataset (5d250693) Test_ , data... +2	1.6min	-	sklearn	0.85572139...	random_forest_classifier
	adaptable-cat-544	4 days ago	dataset (56e86a11) Training_ , c... +2	2.2min	-	sklearn	0.85572139...	xgboost_classifier
	wistful-stag-8	4 days ago	dataset (bfeca8f5) Train , datas... +2	2.2min	-	sklearn	0.85572139...	xgboost_classifier
	dapper-ape-619	4 days ago	dataset (bfeca8f5) Train , datas... +2	2.1min	-	sklearn	0.85572139...	lightgbm_classifier
	clean-bug-44	4 days ago	dataset (b20a8b8c) Val_ , datas... +2	1.7min	-	sklearn	0.85567010...	logistic_regression_classifier
	polite-deer-263	4 days ago	dataset (b20a8b8c) Val_ , datas... +2	2.0min	-	sklearn	0.85567010...	logistic_regression_classifier
	thoughtful-wolf-738	4 days ago	dataset (5d250693) Test_ , data... +2	1.8min	-	sklearn	0.85567010...	logistic_regression_classifier
	dashing-fish-812	4 days ago	dataset (bfeca8f5) Train , datas... +2	1.9min	-	sklearn	0.85427135...	xgboost_classifier
	casual-fish-795	4 days ago	dataset (5d250693) Test_ , data... +2	1.7min	-	sklearn	0.85148514...	random_forest_classifier
	placid-mole-217	4 days ago	dataset (bfeca8f5) Train , datas... +2	2.1min	-	sklearn	0.85148514...	random_forest_classifier
	luminous-kite-848	4 days ago	dataset (5d250693) Test_ , data... +2	1.8min	-	sklearn	0.85148514...	random_forest_classifier
	adorable-stag-649	4 days ago	dataset (5d250693) Test_ , data... +2	1.9min	-	sklearn	0.85148514...	lightgbm_classifier
	silent-rat-198	4 days ago	dataset (56e86a11) Training_ , c... +2	1.8min	-	sklearn	0.85148514...	random_forest_classifier
	amazing-skink-828	4 days ago	dataset (bfeca8f5) Train , datas... +2	2.1min	-	sklearn	0.85148514...	decision_tree_classifier
	grandiose-jay-273	4 days ago	dataset (bfeca8f5) Train , datas... +2	2.0min	-	sklearn	0.85148514...	lightgbm_classifier
	resilient-vole-413	4 days ago	dataset (b20a8b8c) Val_ , datas... +2	2.0min	-	sklearn	0.85148514...	xgboost_classifier

Fig. 3.20: AutoML Experiment Runs for 1,000 Data - Classification

Table 3.4: Comparison of the Result for Classification in Cortex and AutoML - Duration and Costs

	Snowflake	Databricks	Snowflake	Databricks	Snowflake	Databricks
Rows	**1000**		**10000**		**100000**	
Cost	0,03$	0,84$	0,04$	0,84$	0,05$	0,83$
Duration	31s	2h 1m 50s	34s	2h 1m 37s	43s	2h 1m 7s

In conclusion, the comparative analysis of classification performance between Cortex and AutoML reveals differences depending on the dataset size. For smaller datasets of 1,000 and 10,000 data points, Cortex has better performance, though both platforms produce accurate results with Snowflake showing slightly better error metrics. However, for the large dataset of 100,000 data points, AutoML significantly outperforms Cortex, achieving markedly better results. Notably, Snowflake's classification accuracy drops to around 58% for this large dataset, indicating performance issues. The table also compares Cortex and AutoML based on execution duration and costs, revealing that Snowflake consistently delivers better results across all scenarios with similar costs and durations regardless of dataset size. Databricks follows a similar pattern, taking approximately 2 hours per scenario due to a preset timeout. This timeout is critical as Databricks trains numerous models and selects the best one after the 2-hour period, emphasizing the importance of proper timeout configuration. The corresponding figure provides a visual representation of some classification experiments conducted in Databricks.

3.2. DATA ENGINEERING

Snowflake and Databricks both offer robust solutions for data engineering but differ significantly in their approaches to infrastructure management and cost. Snowflake automat-

ically handles all infrastructure, allowing a focus solely on data tasks without concerns about hardware or software configurations. This ease of use extends to its pricing model, where payment is only for the storage and compute resources utilized, ensuring cost efficiency. In contrast, Databricks requires setting up and configuring clusters manually, which can be complex and time-consuming. This manual configuration also means paying for the instances created and run, potentially leading to higher costs and additional management overhead. The next subsections will focus on the description of the operations for data engineering.

3.2.1. Data Engineering - Databricks

The next aspect of the reference architecture is Data Engineering, which primarily highlights basic operations such as preparation, enrichment, pivoting, transformation, aggregation, and wrangling. This chapter will analyze tools for both platforms used for these operations.

Databricks offers a comprehensive platform for data engineering, allowing for a variety of data operations including preparation, enrichment, transformation, aggregation, and data wrangling. The core of Databricks' data engineering capabilities is built around Apache Spark, a powerful, open-source unified analytics engine for large-scale data processing. The main tool used for implementing data engineering operations is Databricks Notebooks, which are already described in the chapter „AI Engineering - Databricks". Notebooks support multiple languages such as Python, Scala, SQL, and R. These notebooks are collaborative and can be used to write, execute, and debug code. They are ideal for iterative data exploration, visualization, and performing complex data transformations.

Below is the description of the data engineering operations mentioned in the reference architecture for Databricks, including examples of their usage [5]:

1. **Data Preparation:**
   - Data Loading: Databricks can connect to numerous data sources, including cloud storage like AWS S3, Azure Blob Storage, and Google Cloud Storage, as well as traditional databases. Data is loaded into Databricks using Spark's DataFrame API.
   - Data Cleaning: Spark's DataFrame transformations are used to handle missing data, remove duplicates, or filter out irrelevant data points. Functions such as .dropna(), .fillna(), and .dropDuplicates() are particularly useful.
2. **Data Enrichment:**
   - Joining Data: Datasets are enriched by joining them with other relevant data using Spark's DataFrame API, which supports various types of joins (inner, outer, left, right).
   - Adding New Columns: SQL queries or DataFrame transformations are used to

create new columns that are derivations or aggregations of existing data, enhancing datasets with additional insights.

3. **Pivoting Data:**
   — Pivot and Unpivot: Databricks supports pivoting data using the groupBy() and pivot() DataFrame functions. This transforms rows into columns, summarizing data for certain types of reporting and analysis.
4. **Data Transformation:**
   — Complex Transformations: Spark SQL or DataFrame APIs are used to transform data, such as splitting strings, changing date formats, or applying any row-wise transformations.
   — UDFs: User-defined functions (UDFs) are written to perform custom transformations on data, which are then run in parallel across the cluster.
5. **Data Aggregation:**
   — Aggregating Data: The DataFrame API is used to perform aggregations like count, sum, average, min, and max. These are combined with groupBy() for more complex aggregational insights.
   — Window Functions: Spark's window functions are leveraged to perform sophisticated aggregations over a specified range of data, such as moving averages or cumulative sums.
6. **Data Wrangling:**
   — Filtering and Sorting: Datasets are easily filtered and sorted using DataFrame operations.
   — Complex Data Manipulations: Complex data wrangling techniques such as dataset reshaping, data type transformations, and applying conditional logic across large datasets are implemented.

3.2.2. Data Engineering - Snowflake

Snowflake provides a complete set of tools for data engineering, designed to efficiently manage data preparation, enrichment, transformation, aggregation, and wrangling. A major aspect of Snowflake's appeal is its scalability, performance, and seamless integration with various data engineering tools and frameworks, such as Apache Spark, Snowpark, and widely used notebook environments. Snowflake integrates with various notebook platforms such as Jupyter, Apache Zeppelin, and Databricks notebooks. This integration is facilitated via Snowflake's connectors (ODBC, JDBC) or directly through specific plugins or extensions designed for these platforms. Notebooks are instrumental in providing an interactive environment for data exploration, visualization, and collaborative data science workflows. Below is the description of the data engineering operations mentioned in the reference architecture for Snowflake, including examples of their usage [27]:

1. **Data Preparation:**
   — Data Loading: Snowflake's bulk loading capabilities via the COPY INTO command are used to import large datasets from cloud storage. For continuous, real-time data ingestion, Snowpipe can be utilized to load data.
   — Data Cleaning: SQL commands are leveraged to perform typical data cleaning tasks such as filtering out invalid data, filling missing values with COALESCE or NVL, and correcting data formats using CAST or TRY_CAST.
2. **Data Enrichment:**
   — Adding Context: Data is enhanced by joining it with additional datasets using Snowflake's powerful JOIN operations. This provides more context for analysis, such as adding geographical information to customer data.
   — Calculated Fields: New columns with calculated values are created using SQL expressions, allowing for deeper insights into the data, such as customer lifetime value or time-based decay models.
3. **Pivoting Data:**
   — PIVOT Function: The PIVOT function is used to transform rows into columns, which is particularly useful for creating summary reports. Snowflake's SQL supports pivoting on specified categories and performing aggregations like sum or average during the pivot.
4. **Data Transformation:**
   — SQL Transformations: SQL is used for transformations, including string manipulations, date calculations, and conditional logic with CASE statements.
   — Window Functions: Window functions like ROW_NUMBER(), LAG(), LEAD(), and cumulative aggregates are implemented to perform complex analytical computations directly in SQL queries.
5. **Data Aggregation:**
   — Aggregation Queries: Aggregation functions like SUM(), AVG(), and COUNT() are utilized, along with GROUP BY clauses to summarize data, making it easier to extract insights and make business decisions.
   — Rollups and Cubes: ROLLUP and CUBE are used for multidimensional aggregations, which are useful for business intelligence and reporting.
6. **Data Wrangling:**
   — Complex Joins: Snowflake supports various types of joins (INNER, LEFT, RIGHT, FULL) which are essential for combining data from multiple sources into a single coherent dataset.
   — Data Type Manipulations: Data types are converted and complex data structures are handled, particularly when working with JSON or other semi-structured data using Snowflake's powerful JSON functions.

3.2.3. Comparison of Data Enigneering

Databricks uses Spark commands, leveraging the power of Apache Spark for distributed data processing, which allows for handling large-scale data transformations and analytics efficiently. This requires developers to be proficient in Spark's APIs, which include languages like Scala, Python, and R. On the other hand, Snowflake relies on SQL commands for data operations, making it more accessible for teams with strong SQL skills. Therefore, the choice between Databricks and Snowflake often depends on the existing skill set of the company's developers. Companies with a team experienced in Spark might prefer Databricks for its advanced capabilities, while those with SQL-savvy developers might lean towards Snowflake for its simplicity and ease of use.

The new addition to Snowflake's data engineering toolkit is Snowpark that allows users to build sophisticated data pipelines directly within Snowflake using familiar programming languages like Scala (and recently Java and Python). It extends the capabilities of Snowflake by enabling complex data processing tasks to be defined and executed as part of Snowflake's compute layer, thus taking advantage of Snowflake's powerful data processing engine. The Snowflake documentation states that with Snowpark, customers experience a median performance improvement of 4.6 times faster and a 35% lower cost compared to managed Spark solutions. On the other hand, Databricks provides a new feature called Databricks SQL, which is a serverless data warehouse based on the lakehouse architecture. Therefore, the conclusion will only be based on the standard solutions of Databricks and Snowflake [27] [5].

In conclusion, Snowflake is generally the better choice for data engineering if a hands-off approach to infrastructure management and a cost-efficient model where payment is only for what is used are preferred. Databricks, while offering greater flexibility and power, involves more effort in cluster management and can lead to higher costs, making it less suitable for those seeking simplicity and cost-effectiveness.

3.3. LAKEHOUSE

In this section, there will be a comparison between the general overview of data lakehouses in Snowflake and Databricks.

3.3.1. Lakehouse in Snowflake

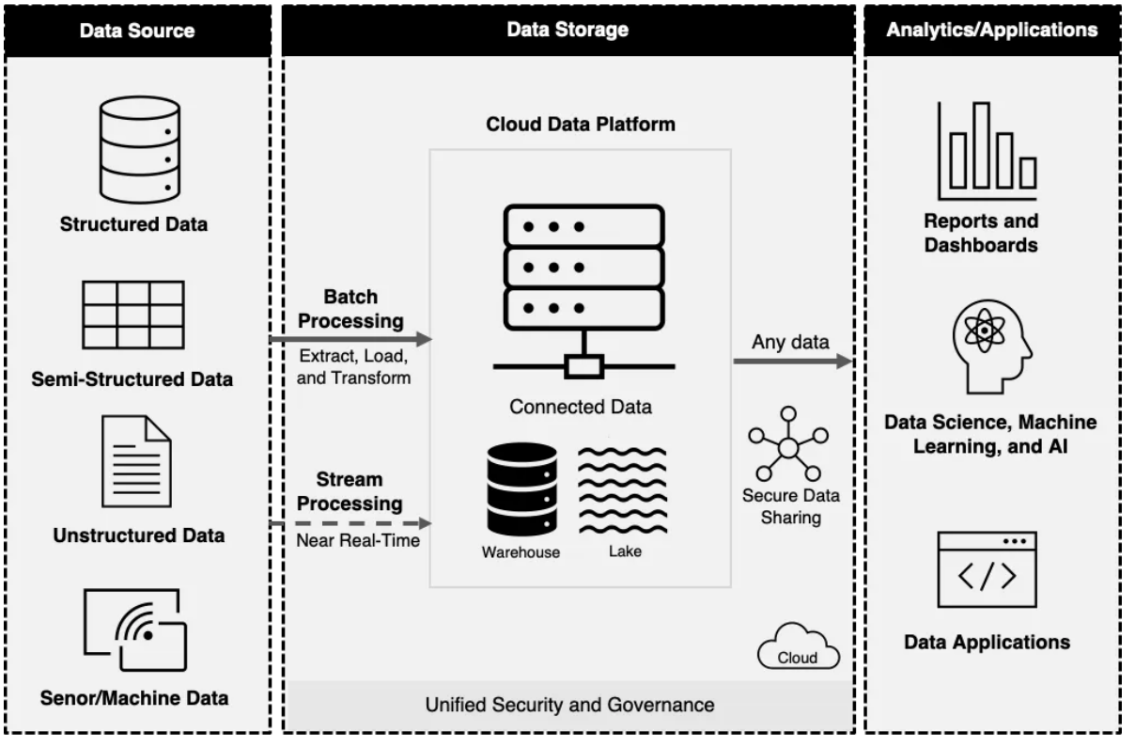


Fig. 3.21: Modern Cloud Data Architecture, which supports Lakehouse [19]

The architecture 3.21 represents a modern data lakehouse setup in Snowflake, highlighting the flow from data ingestion through storage to usage. It showcases the platform's capability to handle diverse data types and complex workflows, emphasizing real-time processing, security, and seamless integration across various data and analytical processes. This kind of architecture supports a wide range of use cases, from operational analytics and BI to advanced machine learning and data sharing, making it highly adaptable to the evolving needs of modern businesses [19]. Below is a description of the data source and data storage stages in the architecture 3.21.

1. **Data Source - this stage emphasizes the diversity of data types that can be ingested into a data lakehouse architecture:**
   — Structured Data: Traditional database records that are highly organized and easily searchable, usually stored in rows and columns like SQL databases.
   — Semi-Structured Data: Includes formats like JSON, XML, and CSV, which may not fit neatly into traditional database tables but still contain tags or markers to separate semantic elements.
   — Unstructured Data: Data that doesn't have a predefined model or format, such as text files, images, videos, etc.
   — Sensor/Machine Data: Data generated from devices or sensors, often used in IoT (Internet of Things) setups, typically ingested in real-time or near real-time.

2. **Data Storage** - this stage outlines the components of the cloud data platform, which serves as the core of the Snowflake Lakehouse:
   — Batch Processing: Traditional ETL (Extract, Load, Transform) processes are used for structured and semi-structured data, suitable for scenarios where real-time processing is not required.
   — Stream Processing: Handles real-time data ingestion, essential for sensor or machine-generated data, reflecting the need for timely data processing and insights.
   — Connected Data: Represents the integration capabilities within the platform, allowing for seamless data flow and connectivity between various storage and processing layers.
   — Warehouse and Lake: Data is stored in optimized formats within a data warehouse for structured and semi-structured data or a data lake for raw, unstructured data.
   — Unified Security and Governance: Critical for managing data access, ensuring compliance, and maintaining data integrity across the platform.
   — Cloud: Indicates that all components are cloud-based, offering scalability, flexibility, and reduced overhead compared to on-premise solutions [27] [19].

3.3.2. Lakehouse in Databricks

Below an architecture diagram 3.22 represents how Databricks uses Delta Lake to unify the capabilities of data lakes and warehouses, thereby creating a highly efficient and scalable lakehouse architecture that supports advanced data operations and analytics. The data source and analytics/applications part in this architecture is very similar to the Snowflake architecture. The core component is Delta Lake, positioned at the center and serving as the foundation.

Delta Lake is an optimized storage layer support tables in a Databricks lakehouse. This open-source software enhances Parquet data files with a file-based transaction log, enabling ACID transactions and scalable metadata management. Fully compatible with Apache Spark APIs, Delta Lake is designed to integrate seamlessly with Structured Streaming, allowing you to use a single data copy for both batch and streaming operations, thereby supporting large-scale incremental processing [5]. Key features of Delta Lake [24]:

— **ACID Transactions:** Ensures data integrity by enabling atomic, consistent, isolated, and durable transactions within big data systems.
— **Scalable Metadata Handling:** Efficiently handles metadata operations even at a massive scale, which traditional data lakes struggle with.
— **Schema Enforcement and Evolution:** Automatically manages and enforces data schema to prevent errors caused by corrupt or inconsistent data. It also allows for schema changes without disruption to downstream processes.

— **Unified Batch and Stream Processing:** Provides a consistent framework for processing both batch and real-time streaming data using the same data infrastructure.

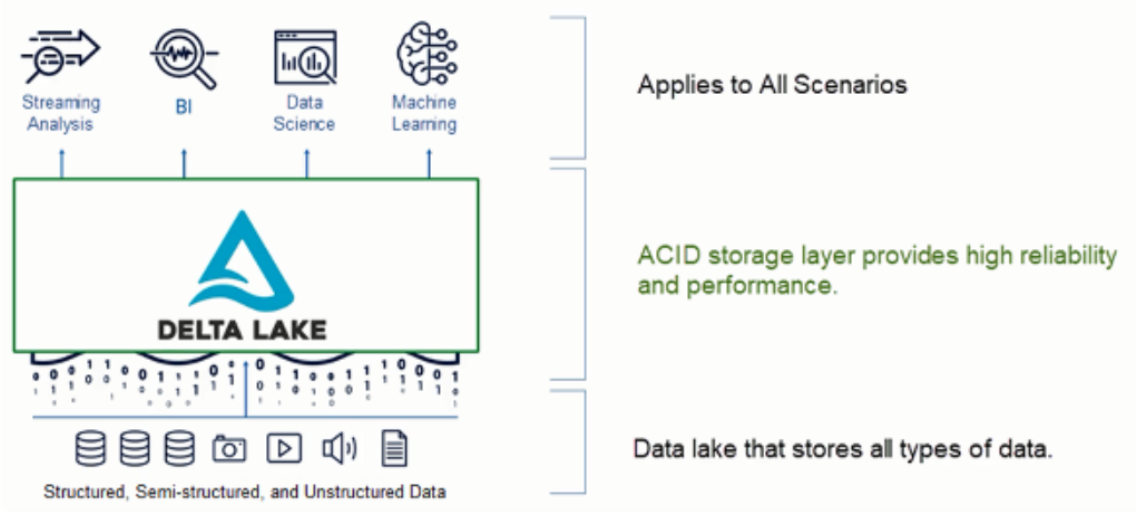


Fig. 3.22: Databricks with Delta Lake architecture [24]

3.3.3. Comparison of Snowflake and Databricks Lakehouse

In the Snowflake ecosystem, while there isn't a direct one-to-one equivalent to Delta Lake since Delta Lake specifically targets problems within the Apache Spark environment, Snowflake incorporates similar functionalities natively within its architecture that address many of the same challenges Delta Lake aims to solve. Here are the features and capabilities in Snowflake that offer similar benefits to those provided by Delta Lake:

— **ACID Transactions:** delta Lake provides ACID transactions to ensure data integrity within big data environments, which is crucial for handling concurrent read and write operations. Similarly, Snowflake supports ACID-compliant transactions within its platform. This means that all operations within Snowflake (inserts, updates, deletes) are fully transactional, providing consistency and data integrity, much like Delta Lake does for data lakes built on Apache Spark.

— **Schema Enforcement and Evolution:** delta Lake allows for schema enforcement to prevent corrupt data from causing issues during data processing. It also supports schema evolution to adapt to changing data sources without causing disruptions. Snowflake handles these through its robust support for semi-structured data types (like JSON) and schema-on-read capabilities. While Snowflake does not require schema enforcement upon data loading, it allows for flexible schema evolution, where users can easily adjust tables and views to accommodate evolving data structures.

— **Time Travel and Data Versioning:** delta Lake provides snapshot isolation, allowing users to access previous versions of the data, which is helpful for auditing, rollbacks, or reproducing experiments. Snowflake offers a similar feature known as Time Travel,

which allows users to access historical data and perform point-in-time analysis. Time Travel can revert data to an earlier state, protect against accidental data loss, and easily manage data modifications.

— **Scalable Metadata Handling:** one of the core benefits of Delta Lake is its ability to handle large amounts of metadata efficiently, which can become problematic in traditional data lakes at scale. Snowflake addresses this challenge inherently with its unique architecture that separates compute, storage, and services. This separation allows Snowflake to manage large datasets and their metadata efficiently without performance degradation.

— **Unstructured Data Support:** while Delta Lake is renowned for improving the handling of structured and semi-structured data in lake environments, Snowflake extends capabilities to unstructured data, which it started supporting recently. This allows Snowflake to store, manage, and analyze unstructured data alongside structured data in a single platform, providing a more holistic approach to data management.

— **Performance Optimization:** both Delta Lake and Snowflake enhance the performance of data operations. Delta Lake achieves this through optimizations like compaction and caching on top of Spark, whereas Snowflake provides automatic performance optimizations like micro-partitioning, caching, and a result cache that speeds up query performance without manual tuning.

In conclusion, while both Snowflake and Databricks enable effective data lakehouse implementations, their approaches and toolsets differ significantly. Snowflake requires combining various available tools to achieve a comprehensive lakehouse solution, leveraging its robust data warehousing capabilities alongside integrations with data lakes and other third-party tools for a cohesive data management and analytics environment. This approach ensures a managed, scalable, and secure setup but may require additional effort to integrate disparate systems. In contrast, Databricks employs Delta Lake as an integrated environment, offering a unified platform for data processing, storage, and analytics. Delta Lake provides built-in support for ACID transactions, scalable metadata handling, and advanced analytics, simplifying the lakehouse implementation within a single, cohesive framework. This integration facilitates real-time data processing and collaborative workflows, making Databricks a good choice for organizations searching flexibility and advanced analytics within a unified environment.

4. DATA FOUNDATION

This chapter provides a detailed comparative analysis of Snowflake and Databricks, with a particular focus on the "Data Foundation" component of reference architecture.

Following the data capture stage, the processes of data validation, including audit, balance, control, and reconciliation, are examined to understand how each platform ensures data integrity and accuracy. The comparison then explores data preprocessing techniques used by Snowflake and Databricks, highlighting their approaches for cleaning and preparing data for further analysis.

The examination extends to the management of data within data lakes, distinguishing between raw (bronze) and cleansed (silver). The analysis assesses how each platform supports these stages of data refinement and integration, evaluating their scalability, performance, and ease of use.

Overall, this chapter aims to uncover the strengths and limitations of Snowflake and Databricks in creating a robust and scalable data foundation, providing insights into their suitability for various data-driven applications and organizational needs. By offering a detailed comparison, this analysis contributes to a deeper understanding of how these platforms support the essential functions of modern data infrastructure.

4.1. DATA CAPTURE

The analysis starts with "Data Capture", looking at how each platform handles stream and batch processing, emphasizing their respective capabilities, efficiencies, and technological methods.

The section will begin with definitions of stream and batch processing, then proceed to compare the available tools and key features for these processing types in Snowflake and Databricks.

Batch processing is a data processing technique where data is collected, processed, and stored in predefined chunks or batches over a period of time. Instead of handling data immediately as it arrives, batch processing waits for a certain amount of data or a scheduled time to process it all at once. This method is particularly useful for tasks that involve large volumes of data, like ETL (Extract, Transform, Load) operations, report generation, and data backups. Batch processing offers advantages such as high throughput, efficient resource utilization, and the ability to handle massive datasets. However, it has the drawback of higher latency, as insights or results are only available after the entire batch is

processed. It is typically well-suited for tasks where real-time processing is not critical and the focus is on optimizing data handling and computational efficiency [3].

Stream processing is a data processing method that handles data in real-time as it is generated. Unlike batch processing, stream processing continuously processes data as it flows, allowing for immediate insights and actions. This approach is ideal for tasks that require real-time analytics, monitoring, and decision-making, such as fraud detection, live dashboard updates, social media sentiment analysis, and IoT (Internet of Things) data processing. Stream processing systems are designed to manage high-velocity data streams with low latency and rapid processing. They often include complex infrastructure and fault-tolerance mechanisms to handle data as it arrives, even if it is out of order or at varying speeds. Stream processing is perfect for applications where timely insights and quick responses to data are crucial [3]. The table 4.1 shows comparison of batch and stream processing.

Table 4.1: Comparison of Batch Processing and Stream Processing [3]

Criteria	**Batch Processing**	**Stream Processing**
Nature of Data	Processed in chunks or batches.	Processed continuously, one event at a time.
Latency	High latency: insights are obtained after the entire batch is processed.	Low latency: insights are available almost immediately or in near-real-time.
Processing Time	Scheduled (e.g., daily, weekly).	Continuous.
Infrastructure Needs	Significant resources might be required but can be provisioned less frequently.	Requires systems to be always on and resilient.
Throughput	High: can handle vast amounts of data at once.	Varies; optimized for real-time but might handle less data volume at a given time.
Complexity	Relatively simpler as it deals with finite data chunks.	More complex due to continuous data flow and potential order or consistency issues.
Ideal Use Cases	Data backups, ETL jobs, monthly reports.	Real-time analytics, fraud detection, live dashboards.
Error Handling	Detected after processing the batch; might need to re-process data.	Needs immediate error-handling mechanisms; might also involve later corrections.
Consistency & Completeness	Data is typically complete and consistent when processed.	Potential for out-of-order data or missing data points.
Tools & Technologies	Hadoop, Apache Hive, batch-oriented Apache Spark.	Apache Kafka, Apache Flink, Apache Storm.

Additionally, the comparison will include definitions of Catalyst Optimizer and Tungsten Execution Engine.

Tungsten is the codename for a comprehensive project dedicated to upgrading Apache Spark's execution engine. Its primary goal is to significantly improve memory and CPU efficiency for Spark applications, pushing performance to the limits of contemporary hardware [5]. Catalyst Optimizer is a crucial component of Apache Spark SQL, aimed at optimizing Spark SQL query execution. Introduced in Spark 1.2, Catalyst utilizes various advanced optimization techniques to minimize query execution time, thereby enabling Spark to analyze, organize, and execute tasks more effectively [25].

4.1.1. Data Capture in Snowflake and Databricks

Both Databricks and Snowflake offer robust environments for handling batch and stream processing, albeit through different technological paradigms and architectural approaches. Here's a more detailed exploration:

1. **Batch Processing in Databricks:** databricks leverages Apache Spark for batch processing, providing a highly scalable, distributed computing environment. Batch processing in Databricks is designed for handling large volumes of data efficiently across a distributed set of resources in the cloud [5].
   — **Scalability:** utilizes Spark's ability to distribute data across clusters and process in parallel, making it ideal for big data applications.
   — **Optimization:** spark's Catalyst optimizer and Tungsten execution engine optimize physical execution plans and generate efficient bytecode, respectively, to speed up processing.
   — **Data Integration:** supports diverse data sources, including HDFS, S3, Azure Data Lake, Kafka, Cassandra, and more, with the capability to handle complex data types and formats seamlessly.
2. **Stream Processing in Databricks:** for stream processing, Databricks integrates Spark Structured Streaming. This API allows for fast, easy, and fault-tolerant stream processing of live data streams, aligning closely with batch processing to handle complex, real-time analytics [5].
   — **Unified API:** structured Streaming provides a high-level API that integrates with Spark's batch APIs, simplifying development by treating stream processing as an extension of batch processing.
   — **Event-time processing:** supports event-time and late data handling, watermarking, and window functions, enabling sophisticated time-series analyses.
   — **Fault Tolerance:** uses Spark's resilient distributed datasets (RDDs) and checkpointing to provide strong fault tolerance guarantees against data loss.
3. **Batch Processing in Snowflake:** Snowflake processes batch jobs using its SQL engine, which separates compute from storage, allowing for flexible, on-the-fly scaling of

resources without downtime. Its architecture supports massive parallel processing (MPP), making it highly efficient for large-scale batch operations [27].

— **Performance:** virtual warehouses can be scaled up or down to provide more or less compute power as needed for batch processing demands.

— **Zero-Copy Cloning:** offers the ability to clone data without additional storage costs, facilitating efficient testing and development on large datasets.

— **Concurrency:** manages and scales automatically to handle multiple batch jobs simultaneously without performance degradation, thanks to its multi-cluster shared architecture.

4. **Stream Processing in Snowflake:** While Snowflake does not natively support real-time stream processing like Databricks, it handles near-real-time data loads through Snowpipe. Snowpipe listens for notification of new data files uploaded to cloud storage and loads data continuously into Snowflake [27].

— **Snowpipe:** uses REST APIs to automate data loading from cloud storage, enabling near-real-time processing by continuously fetching and loading data as it arrives.

— **Micro-batching:** to achieve real-time streaming, Snowflake integrates with external tools like Apache Kafka, Amazon Kinesis, or third-party services, which can push data to Snowflake for processing.

— **Integration with External Tools:** manages and scales automatically to handle multiple batch jobs simultaneously without performance degradation, thanks to its multi-cluster shared architecture.

4.1.2. Comparison of Data Capture

Table 4.2: Detailed Comparison of Databricks and Snowflake on Batch and Stream Processing

Aspect	Databricks	Snowflake
Batch Processing Scalability	High scalability with Spark, supporting parallel processing across clusters.	Virtual warehouses can be scaled dynamically to meet batch processing demands.
Batch Optimization	Utilizes Catalyst optimizer and Tungsten execution engine for efficient execution plans and bytecode generation.	Uses advanced query optimization techniques to efficiently manage and execute batch workloads.
Data Integration	Integrates seamlessly with a variety of data sources such as HDFS, S3, Azure Data Lake, Kafka, and Cassandra.	Supports a wide range of data connectors and formats, with efficient data handling via external tables and native integrations.
Stream Processing Capability	Offers Spark Structured Streaming for tightly integrated real-time data processing.	Utilizes Snowpipe for near-real-time ingestion, enhanced by external tools for complete streaming capabilities.
Event-Time Processing	Supports complex event-time processing with watermarking and window functions.	External tools required for similar capabilities, as Snowflake's native features focus more on batch-oriented workflows.
Fault Tolerance in Streaming	Strong fault tolerance provided by Spark's resilient distributed datasets (RDDs) and checkpointing.	Relies on continuous data loading mechanisms and external stream processing tools for fault tolerance.
Use Case Adaptability	Excellently suited for real-time analytics, complex event detection, and dynamic data pipeline constructions.	Optimized for large-scale batch processing, real-time data ingestion for specific cases, and operational analytics.

The table 4.2 provides a clear distinction between how Databricks and Snowflake approach batch and stream processing. Databricks utilizes Apache Spark to offer a robust, integrated solution that handles both batch and stream data efficiently within the same environment, making it highly suitable for applications requiring intensive, real-time data

processing alongside traditional batch workflows. Snowflake, on the other hand, excels in batch processing with its powerful SQL engine and innovative data loading tool, Snowpipe, but relies on external integrations for real-time stream processing capabilities, positioning it as a flexible solution for organizations with strong batch processing needs and less frequent real-time requirements.

The above comparison is only for traditional tools of Snowflake, based on SQL, and Databricks, based on Spark. There are also important new features, namely Snowpark and Delta Lake. Delta Live Table (DLT) simplifies batch processing through a declarative pipeline approach and supports streaming data natively, allowing users to define continuous data pipelines with minimal configuration. On the other hand, Snowflake allows developers to write data processing code in their preferred language, using a DataFrame API similar to Spark. It is very flexible for user needs but requires more manual setup compared to DLT. In conclusion, Databricks provides a more integrated and seamless experience for both batch and stream processing, particularly with Delta Live Tables. Snowflake Snowpark offers powerful batch processing capabilities with a flexible programming model, but stream processing requires more manual intervention and is better suited for near real-time use cases.

4.2. MEDALLION ARCHITECTURE

This section will analyze and compare the implementation of medallion architecture in Snowflake and Databricks. It will begin with a definition of medallion architecture and an explanation of data at all stages.

A medallion architecture acts as a data design framework specifically crafted for organizing data in a lakehouse setting. Its main goal is to progressively improve the structure and quality of data as it moves through each layer of the architecture, transitioning from Bronze to Silver to Gold layers [23]. Below is a description of these layers [23]:

— **Bronze Layer:** the bronze layer is the initial landing zone for all data from external source systems. Datasets in this layer retain the structure of the source system tables in their original form, supplemented with additional metadata columns such as load date/time and process ID. The primary focus here is on Change Data Capture, enabling historical archiving of source data, maintaining data lineage, facilitating audit trails, and allowing for reprocessing if necessary without needing a fresh read from the source system.

— **Silver Layer:** the next layer in the lakehouse is the Silver layer. Here, data from the Bronze layer undergoes a series of operations to reach a "just-enough" state, preparing it to provide a comprehensive "enterprise view" of essential business entities, concepts, and transactions.

— **Gold Layer:** the final layer in the lakehouse is the Gold layer. Data in this layer is typically organized into subject area-specific databases, optimized for consumption. This layer is focused on reporting, utilizing denormalized, read-optimized data models with minimal joins. It represents the final stage for applying data transformations and quality rules. The following diagram 4.1 represents medallion architecture:

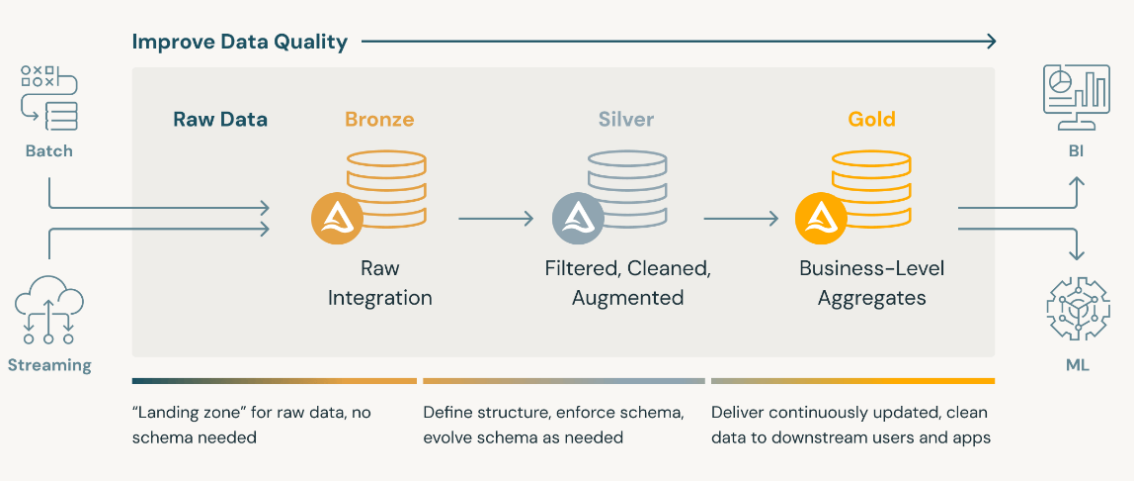


Fig. 4.1: Medallion Architecture [5]

The following subsections will analyze and compare the implementation of medallion architecture in Databricks and Snowflake.

4.2.1. Medallion Architecture in Snowflake

In Snowflake, there is no specific, integrated environment for medallion architecture, but there is a possibility to achieve this using the following tools for particular data stages [27] [15]:

1. **Bronze Layer:**
   — **Bulk Loading:** Snowflake supports efficient bulk loading of data through the COPY INTO command, which can rapidly import large amounts of data from files stored in cloud storage services like Amazon S3, Google Cloud Storage, or Azure Blob Storage.
   — **External Tables:** Snowflake can query data directly from external sources using external tables. This enables users to perform SQL queries on data that resides in external storage without fully ingesting it into Snowflake, reducing storage costs and ingestion times.
   — **Snowpipe** - see Section 4.1
2. **Silver Layer:**
   — **Transformation** - see Section 3.2.2

— **Streams and Tasks:** Snowflake supports the use of Streams to capture changes to tables and Tasks to automate the transformation processes. This helps in managing data flows more effectively and ensures that data is up-to-date and ready for further analysis.
— **Zero-Copy Cloning** - see Section 4.1

3. **Gold Layer:**
— **Materialized Views:** Snowflake supports the creation of Materialized Views to store pre-computed results, which can speed up query performance for complex analytics tasks frequently run against Gold layer data.
— **Secure Views and Data Sharing:** to facilitate secure and governed access to Gold layer data, Snowflake allows the creation of Secure Views that restrict data access based on user roles. Additionally, Snowflake's unique Data Sharing capabilities enable organizations to share selected datasets with other Snowflake users without duplicating data.
— **Scalable Compute:** the use of separate compute clusters (virtual warehouses) can be scaled according to the analytic demands of the Gold layer. This ensures that resources are efficiently allocated, optimizing performance and controlling costs.

4.2.2. Medallion Architecture in Databricks

Delta Live Tables (DLT) is a declarative ETL framework built for the Databricks Data Intelligence Platform. It helps data teams simplify both streaming and batch ETL processes in a cost-effective way. By just defining the necessary transformations on your data, DLT pipelines take care of everything automatically. They handle task orchestration, manage clusters, monitor activities, ensure data quality, and manage error handling seamlessly [5]. The architecture 4.2 below shows how the medallion architecture is implemented using Delta Live Tables in Delta Lake and illustrates the entire data workflow from start to end.

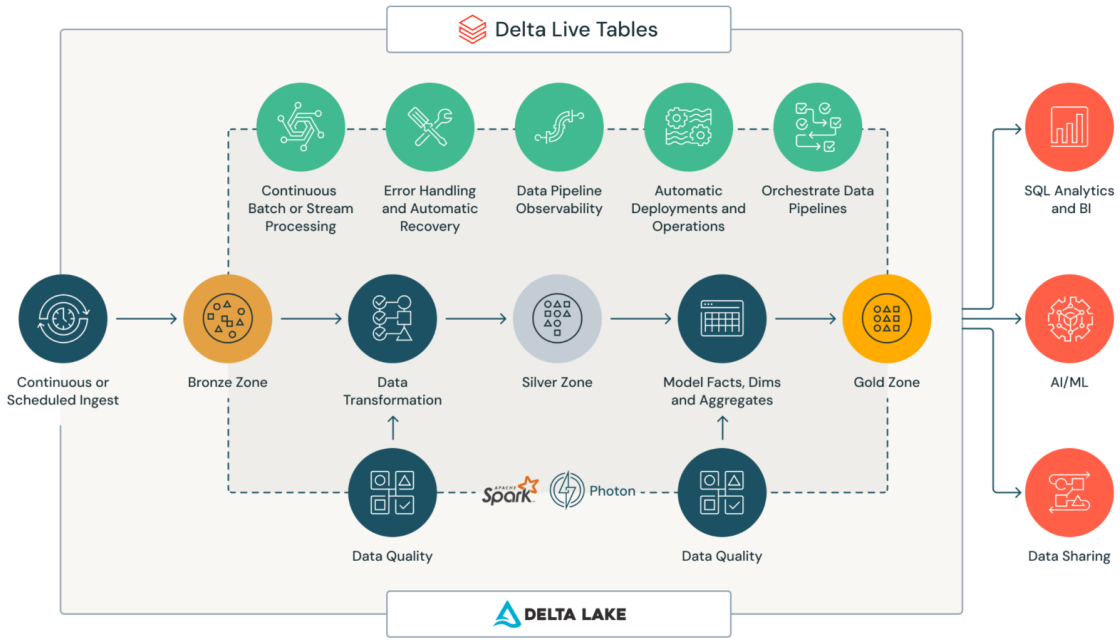


Fig. 4.2: Medallion Architecture in Delta Lake [5]

4.2.3. Plan for Conducting the Research - Medallion Architecture

In the previous sections, there was a description of how to implement the medallion architecture in Snowflake and Databricks. In the next section, there will be a comparison of the methods for implementing the workflow in the bronze, silver, and gold layers, as well as a comparison of performance and costs in Snowflake and Databricks. In Databricks, the implementation of the medallion architecture will not use the DLT pipeline but will directly populate tables using Spark in the notebook. The configuration of the environments is the same as for the AI Engineering experiments described in Section 3.1.3.

To compare the implementation of medallion architecture, similar to AI engineering experiments that generate test data. Investigate how Snowflake and Databricks can handle data by generating 1 million test data entries with columns that can be transformed and enriched for testing purposes. Below is a description of the columns of the generated data:

— id: unique identifier for each record, generated as a UUID (Universal Unique Identifier).
— name: randomly generated name, which can be either male or female.
— randomly generated integer representing the age of the individual, ranging from 18 to 80 years.
— city: randomly chosen city from a predefined list of major European cities.
— purchase_amount: randomly generated floating-point number representing the amount of a purchase, ranging from 10 to 10,000.
— purchase_date: randomly generated date representing the purchase date, ranging from one year ago up to today.

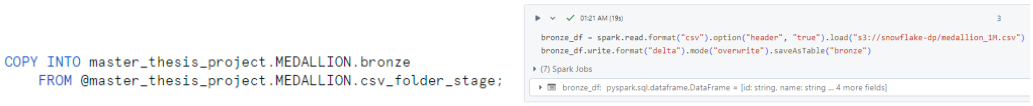
1 million generated data, described in the previous section, were saved to a CSV file and uploaded to an S3 bucket on Amazon Web Services (AWS). The experiment was conducted as follows, in both Snowflake and Databricks:

- **Bronze Layer:**
  - Load raw data from a CSV file from AWS S3 bucket.
  - Save the data to the "bronze" table.
- **Silver Layer:**
  - Perform transformations such as handling missing values, calculating age groups, and flagging high-value purchases.
  - Save the data to the "silver" table.
- **Gold Layer:**
  - Aggregate data to get total and average purchases per customer and segment customers.
  - Save the data to the "gold" table.

Similar to experiments for AI engineering, all of the generated data is available in the public repository [21].

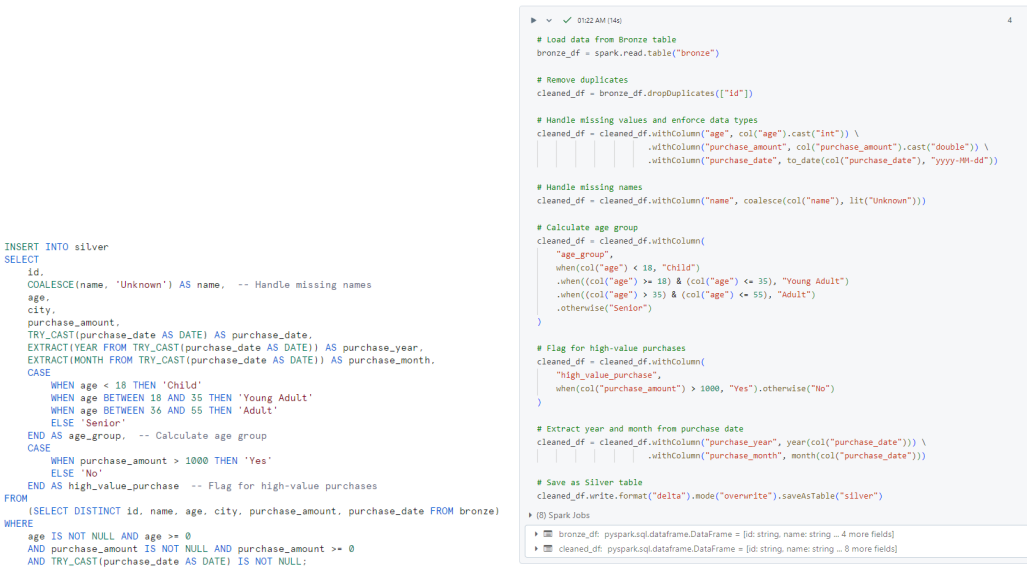
4.2.4. Comparison of Implementation of Medallion Architectures

Below is a comparison between the implementation of the medallion architecture in Snowflake and Databricks for the bronze (Fig. 4.3, silver (Fig. 4.4), and gold (Fig. 4.5) stages:



(a) (b)

Fig. 4.3: Comparison of Bronze Layers in Snowfake (a) and Datarbicks (b)



```sql
INSERT INTO silver
SELECT
    id,
    COALESCE(name, 'Unknown') AS name,  -- Handle missing names
    age,
    city,
    purchase_amount,
    TRY_CAST(purchase_date AS DATE) AS purchase_date,
    EXTRACT(YEAR FROM TRY_CAST(purchase_date AS DATE)) AS purchase_year,
    EXTRACT(MONTH FROM TRY_CAST(purchase_date AS DATE)) AS purchase_month,
    CASE
        WHEN age < 18 THEN 'Child'
        WHEN age BETWEEN 18 AND 35 THEN 'Young Adult'
        WHEN age BETWEEN 36 AND 55 THEN 'Adult'
        ELSE 'Senior'
    END AS age_group,  -- Calculate age group
    CASE
        WHEN purchase_amount > 1000 THEN 'Yes'
        ELSE 'No'
    END AS high_value_purchase  -- Flag for high-value purchases
FROM
    (SELECT DISTINCT id, name, age, city, purchase_amount, purchase_date FROM bronze)
WHERE
    age IS NOT NULL AND age >= 0
    AND purchase_amount IS NOT NULL AND purchase_amount >= 0
    AND TRY_CAST(purchase_date AS DATE) IS NOT NULL;
```

```python
# Load data from Bronze table
bronze_df = spark.read.table("bronze")

# Remove duplicates
cleaned_df = bronze_df.dropDuplicates(["id"])

# Handle missing values and enforce data types
cleaned_df = cleaned_df.withColumn("age", col("age").cast("int")) \
                       .withColumn("purchase_amount", col("purchase_amount").cast("double")) \
                       .withColumn("purchase_date", to_date(col("purchase_date"), "yyyy-MM-dd"))

# Handle missing names
cleaned_df = cleaned_df.withColumn("name", coalesce(col("name"), lit("Unknown")))

# Calculate age group
cleaned_df = cleaned_df.withColumn(
    "age_group",
    when(col("age") < 18, "Child")
    .when((col("age") >= 18) & (col("age") <= 35), "Young Adult")
    .when((col("age") > 35) & (col("age") <= 55), "Adult")
    .otherwise("Senior")
)

# Flag for high-value purchases
cleaned_df = cleaned_df.withColumn(
    "high_value_purchase",
    when(col("purchase_amount") > 1000, "Yes").otherwise("No")
)

# Extract year and month from purchase date
cleaned_df = cleaned_df.withColumn("purchase_year", year(col("purchase_date"))) \
                       .withColumn("purchase_month", month(col("purchase_date")))

# Save as Silver table
cleaned_df.write.format("delta").mode("overwrite").saveAsTable("silver")
```

(a) (b)

Fig. 4.4: Comparison of Siler Layers in Snowfake (a) and Datarbicks (b)

```sql
INSERT INTO gold
SELECT
    id,
    name,
    city,
    COUNT(id) AS total_purchases,
    SUM(purchase_amount) AS total_purchase_amount,
    AVG(purchase_amount) AS average_purchase_amount,
    CASE
        WHEN SUM(purchase_amount) > 5000 THEN 'High Value'
        WHEN SUM(purchase_amount) <= 5000 AND SUM(purchase_amount) > 1000 THEN 'Medium Value'
        ELSE 'Low Value'
    END AS customer_segment
FROM
    silver
GROUP BY
    id, name, city;
```

```python
# Load data from Silver table
silver_df = spark.read.table("silver")

# Aggregate data to get total and average purchases per customer
customer_agg_df = silver_df.groupBy("id", "name", "city").agg(
    count("id").alias("total_purchases"),
    sum("purchase_amount").alias("total_purchase_amount"),
    avg("purchase_amount").alias("average_purchase_amount")
)

# Segment customers based on their total purchase amount
customer_agg_df = customer_agg_df.withColumn(
    "customer_segment",
    when(col("total_purchase_amount") > 5000, "High Value")
    .when((col("total_purchase_amount") <= 5000) & (col("total_purchase_amount") > 1000), "Medium Value")
    .otherwise("Low Value")
)

# Save as Gold table for customer aggregates
customer_agg_df.write.format("delta").mode("overwrite").saveAsTable("gold")
```

(a) (b)

Fig. 4.5: Comparison of Gold Layers in Snowfake (a) and Datarbicks (b)

The table 4.3 shows the duration of the execution stages whose implementations are shown in the above figures. For each stage, Snowflake had better performance. The table 4.4 represents the total duration and total cost of the entire workflow. The total duration in Snowflake is almost three times shorter than in Databricks, but the total cost is three times higher.

Table 4.3: Comparison of the Result for Implementation of Medallion Architecture

	Snowflake	Databricks	Snowflake	Databricks	Snowflake	Databricks
Rows	Bronze Layer		Silver Layer		Gold Layer	
Duration	7s	19s	5s	14s	3s	9s

Table 4.4: Comparison of the Result for Implementation of Medallion Architecture - Total Duration and Cost

	Snowflake	Databricks
Total Duration	15s	42s
Total Cost	0,016$	0,005$

4.2.5. Comparison of Medallion Architectures

Databricks, for the implementation of medallion architecture, uses Delta Live Tables. This architecture allows efficient data processing and management, ensuring high data quality across stages: bronze, silver, and gold. Snowflake can achieve medallion architecture by using a combination of built-in features and functionalities, making this solution very flexible. However, Databricks provides a unified, integrated environment for implementing the medallion architecture through its Delta Live Tables feature. This integration simplifies the process, making it more efficient and user-friendly compared to Snowflake. Delta Live Tables in Databricks streamline data management, automate data pipelines, and enhance data reliability. This approach reduces complexity and allows for easier implementation and maintenance of the medallion architecture, offering a clear advantage over Snowflake's separate components.

The experiment showed that Snowflake has better performance but is also more expensive. The comparison is based on Spark and SQL solutions on both platforms, and it is worth highlighting that Databricks also supports DLT pipelines, which are easy to connect with the implementation code presented in the experiment and easier to manage. On the other hand, Snowflake has tools like Snowpipe, but it requires more configuration. Both platforms worked well for the workflow of 1 million data entries, but based on SQL and Spark solutions, Snowflake is a better choice.

In conclusion, both Databricks and Snowflake effectively implement the medallion architecture, but their distinct methods provide flexibility in choosing the best fit for specific organizational needs and preference, but general conclusion is that Databricks proves to be a better choice for organizations, that want an efficient and simple solution for medallion architecture implementation. However, the experiment proved that Snowflake is a better choice for organizations that want better performance.

4.3. DATA VALIDATION

In this section, there will be a comparison based on data validation. According to the reference architecture, the focus will primarily be on audit, balance, control, and

reconciliation. First, there will be an analysis of the available tools in Databricks and Snowflake, followed by a comparison between them.

4.3.1. Data Validation in Databricks

Data validation ensures the accuracy and quality of data. Below is a description of the available tools and functionalities for data validation in Databricks [5]:

— **Constraints:** Constraints allow the analytics platform to detect data sets with errors and stop them from being added to a table. Delta tables support two types of constraints: NOT NULL - ensures that no NULL values can be inserted into the column and CHECK, which ensures that each input row meets the specified boolean condition.
— **Validate:** The VALIDATE function in Databricks, used in conjuntion with the COPY INTO command, performs a series of checks on the data before the actual copy operation is exectuted. These validation checks help to ensure data quality and consistency by verifyng the following:
  — NOT NULL and CHECK Constraints ensures that all NOT NULL constraints are satisfied and any CHECK constraints defined on the table are met.
  — Data Parsing confirms that the data can be correctly parsed according to the expected format.
  — Schema Matching verifies that the schema of the incoming data set matches the schema of the target table. This ensures that the data structure is consistent and compatible with the table into wich it is being copied.
— **Expectations With DLT (Delta Live Tables):** DLT use validation and integrity checks with declarative quality expectations to stop bad data from passing through data pipelines. Optional ON VIOLATION clause define the handling of data when a constraint is breached.
— **Schema Enforecment:** also known as Schema Validation, is a feature of Delta Lake that stops users and processes from adding data incompatible with the target table's schema.
— **Schema Evloution and Schema Overwrite:** Schema Evolution is allowing the automatic modify of a table's schema for fit new columns in incoming datasets. This process ensure that data practitioners not need rewrite any transformation logic when schema is changes. When changes are need that Schema Evolution cannot handle — such as dropping columns, changing data types, or renaming columns — Schema Overwrite is use. Overwriting the schema can fix these more complex modifications without problems of deleting and recreating the table.
— **Explicit Schema Update:** manually updating a table's schema provides high control over the process. Possible schema alterations include these operations on columns:

adding, changing comment, changing ordering, replacing, renaming, dropping, modifying type.

— **Data Profiling:** Databricks provides a built-in data profiling feature directly within the Databricks Notebooks. Users running clusters with DBR 9.1 or newer can generate data profiles using the cell output UI .

— **Audit:** Databricks records detailed audit logs that monitor user actions, such as data access and modifications. These logs are saved in cluster event logs and can be exported for analysis. Delta Lake offers audit capabilities via its transaction log, which tracks all data changes, enabling the auditing of historical data states.

4.3.2. Data Validation in Snowflake

Below is a description of the available tools and functionalities for data validation in Snowflake [27]:

— **Constraints:** The following constraint types from the ANSI SQL standard are supported by Snowflake: UNIQUE, PRIMARY and FOREIGN KEY, NULL.

— **Validate:** Snowflake, similar to Databricks, supports the VALIDATE function in the COPY INTO command.

— **Stored Procedures and UDFs:** Snowflake supports stored procedures and user-defined functions to implement complex validation logic.

— **Streams and Tasks:** Snowflake Streams capture changes in data, and Tasks automate validation processes by triggering validation checks.

— **Data Profiling:** SQL queries can be used with Snowflake's tools to profle data. By examining the output, it is posible to detect anomalies, outliers, and pattrns that signal data quality issues.

— **Audit:** Snowflake provides comprehensive audit logging capabilities via the ACCOUNT_USAGE and INFORMATION_SCHEMA views. These logs capture detailed information about user activities, queries, data modifications, and access patterns and also Snowflake's Time Travel feature allows users to access historical data states, enabling them to query, clone, and restore data as it existed at any point within a defined retention period.

4.3.3. Comparison of Data Validation

The following table 4.5 compares the data validation tools available in Snowflake and Databricks.

Table 4.5: Detailed Comparison of Data Validation Features in Databricks and Snowflake

Feature	Databricks	Snowflake
Constraints	- NOT NULL, CHECK constraints	- UNIQUE, NOT NULL constraints
Validation	- VALIDATE function with COPY INTO command	- VALIDATE function with COPY INTO command
Schema Management	- **Schema Enforcement:** Prevents incompatible data - **Schema Evolution:** Automatic schema modification - **Schema Overwrite:** Handles complex schema changes - **Explicit Schema Update:** Manual schema updates - **Data Profiling:** Built-in feature in Databricks Notebooks	- **Stored Procedures and UDFs:** Complex validation logic - **Streams and Tasks:** Automates validation processes and captures data changes - **Data Profiling:** Uses SQL queries for data profiling
Data Pipelines	- **Delta Live Tables (DLT):** Validation and integrity checks with declarative quality expectations	- **Streams and Tasks:** Automates validation checks and captures data changes
Audit Capabilities	- Detailed audit logs - Delta Lake transaction logs for tracking data changes	- Comprehensive audit logging in ACCOUNT_USAGE and INFORMATION_SCHEMA - Time Travel for historical data access
Automation and Integration	- DLT for real-time validation and data quality management	- **Streams and Tasks:** for automated validation and data change capture

In summary, both Databricks and Snowflake offer comprehensive data validation solutions, each with unique strengths. Databricks excels in automated schema management and real-time validation via DLT. On the other hand, Snowflake stands out with its strict adherence to ANSI SQL standards, robust constraint options, and detailed audit logging capabilities. The choice between these two platforms often depends on specific project needs, existing infrastructure, and preferred workflow management practices. Thus, deciding which one to use will depend on needs for project. However, as presented in the table 4.5, schema management in Databricks has more built-in features and support functionalities. Therefore, if this aspect is important for the customer, Databricks will be the better option in this case.

5. USER EXPERIENCE

This chapter will cover the general user experience and findings during the analysis and experiments for this thesis conducted in Snowflake and Databricks. The descriptions for these findings are provided below.

The documentation provided by Databricks was notably more readable and easier to navigate and search for information. It included a lot of general information, which was valuable for constructing the background chapter of the thesis. This comprehensive documentation facilitated a deeper understanding of the technologies and methodologies support Databricks.

In terms of user interface (UI), Databricks offered a more extended and feature-rich environment compared to Snowflake. The variety of different views and tools available in Databricks provided a flexible workspace for conducting various data operations, making it highly adaptable for complex workflows. Conversely, Snowflake's UI, while simpler, was exceptionally easy to navigate. Its straightforward design ensured that even new users could quickly familiarize themselves with the platform, making it an accessible option for data analysis tasks.

Operational efficiency also emerged as a significant point of comparison. Snowflake demonstrated better performance in this regard, with tasks executing almost instantly and query details displaying immediately. This immediate feedback loop significantly boosted productivity, especially for iterative data analysis and debugging. In contrast, working with Databricks often involved waiting periods, particularly at the start of a session. The necessity to wait approximately 5-10 minutes for the cluster to start (Fig. 5.1), along with longer execution times for certain operations, highlighted a performance gap between the two platforms.

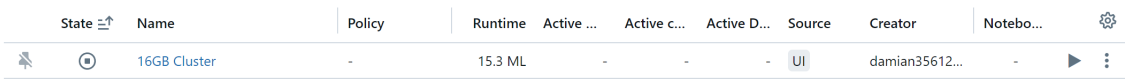


Fig. 5.1: Compute View in Databricks Console

During the experiments, Databricks automatically generated plots for the results, significantly simplifying the analysis process. This automated feature allowed for immediate visualization of data, facilitating quicker insights and streamlined workflows. In contrast, using Snowflake required additional steps to set up and integrate with external tools for visualization. This added complexity made the process more cumbersome and

time-consuming, as users had to rely on third-party tools to visualize and interpret their data. Consequently, Databricks' built-in plotting capabilities provided a more efficient and user-friendly experience for data analysis.

SUMMARY

In this thesis, a comprehensive comparison between Snowflake and Databricks was conducted based on reference architecture (1.6). The comparison included an analysis of available tools and functionalities, as well as practical experiments. Below is a summary of the conclusions from each section.

In the forecasting comparison the analysis between Databricks and Snowflake reveals that for smaller datasets (1,000 and 10,000 rows), both platforms perform well, with Databricks having slightly better error metrics and minimal cost differences. However, for larger datasets (100,000 rows), Snowflake significantly outperforms Databricks in forecasting accuracy and execution efficiency, completing tasks faster and more cost-effectively. Databricks struggles with large datasets, failing to detect patterns efficiently and incurring long execution times due to a 10-hour timeout. Thus, while Databricks is suitable for smaller tasks, Snowflake is the preferable choice for large-scale data forecasting.

For classification the comparative analysis between Cortex and AutoML shows that for smaller datasets (1,000 and 10,000 data points), Cortex performs better, though both produce accurate results. For large datasets (100,000 data points), AutoML significantly outperforms Cortex. Snowflake's classification accuracy drops to around 58% for large datasets, indicating performance issues. Execution duration and cost comparisons reveal that Snowflake delivers better results across all scenarios with similar costs and durations. Databricks takes approximately 2 hours per scenario due to a preset timeout, emphasizing the importance of proper timeout configuration. Therefore, Snowflake is a better choice for smaller datasets in classification tasks, whereas Databricks excels with large datasets, offering robust processing power and advanced analytics suitable for big data environments.

Based on the data engineering aspect, Snowflake is the better choice for those preferring hands-off infrastructure management and a cost-efficient, pay-as-you-go model. Databricks offers more flexibility and power but requires more effort in cluster management and can be more costly, making it less suitable for those seeking simplicity and cost-effectiveness.

Considering implementation of data lakehouse, Snowflake and Databricks both support effective data lakehouse implementations but with different approaches. Snowflake combines various tools to create a comprehensive solution, leveraging its strong data warehousing and third-party integrations, which can require additional integration efforts. This makes Snowflake better suited for organizations that prefer to implement a custom architecture rather than using a single unified framework. In contrast, Databricks uses Delta Lake as an integrated environment, offering a unified platform for data processing,

storage, and analytics, with built-in support for ACID transactions and advanced analytics, making it ideal for organizations seeking flexibility and advanced capabilities within a single framework.

Databricks, leveraging Apache Spark, offers a robust, integrated solution for both batch and stream data processing within the same environment, making it ideal for applications needing intensive real-time data processing and traditional batch workflows. Snowflake excels in batch processing with its powerful SQL engine and Snowpipe for data loading but requires external integrations for real-time stream processing, making it suitable for organizations with strong batch processing needs and infrequent real-time requirements. Databricks' Delta Live Tables simplifies batch processing and supports streaming data natively, providing a seamless experience. In contrast, Snowflake's Snowpark offers powerful batch processing with a flexible programming model, though stream processing requires more manual setup and is better for near real-time use cases.

The summary for implementation medallion architecture, both platforms effectively implement the medallion architecture, offering flexibility to fit specific organizational needs and preferences. However, Databricks generally proves to be a better choice for organizations seeking an efficient and straightforward solution for medallion architecture implementation. However, the experiment proved that Snowflake is a better choice for organizations that want better performance.

For data validation both Databricks and Snowflake offer comprehensive data validation solutions with unique strengths. Databricks excels in automated schema management and real-time validation via Delta Live Tables (DLT), while Snowflake stands out with strict adherence to ANSI SQL standards, robust constraint options, and detailed audit logging capabilities. The choice between these platforms depends on specific project needs, existing infrastructure, and preferred workflow management practices. However, Databricks has more built-in features and support functionalities for schema management, making it the better option if this aspect is important to the customer.

In summary, both Databricks and Snowflake are excellent platforms, each with its strengths. The choice between them should be based on specific organizational requirements and focus on the aspect that is most important to the organization. This thesis covered the "Data as Products Engineering" and "Data Foundation" components from the reference architecture. Future work will focus on the other aspects in the reference architecture, which are "Data Trust", "Platform Foundation", "Insights Activation" and "Collaborative Data Ecosystem".

BIBLIOGRAPHY

[1] Allam, K., Ankam, M., Nalmala, M., Scholar i, R., *Cloud data warehousing: How snowflake is transforming big data management*, INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY. 2023, tom 14, str. 156–162.

[2] Apache Software Foundation, *Apache spark documentation*, `https://spark.apache.org/docs/latest/`. 2024.

[3] Atlan, *Batch processing vs stream processing: 7 key differences*, `https://atlan.com/batch-processing-vs-stream-processing/`. 2024.

[4] Dageville, B., Cruanes, T., Zukowski, M., Antonov, V., Avanes, A., Bock, J., Claybaugh, J., Engovatov, D., Hentschel, M., Huang, J., i in., *The snowflake elastic data warehouse*, w: *Proceedings of the 2016 International Conference on Management of Data* (Springer Berlin Heidelberg, 2016), str. 215–226.

[5] Databricks, Inc., *Databricks documentation*, `https://docs.databricks.com/en/index.html`. 2024.

[6] Dibouliya, A., Bank, W., Stamford, C., *Review on: Modern data warehouse & how is it accelerating digital transformation*, International Journal of Advance Research, Ideas and Innovations in Technology. 2023.

[7] George, D.A.S., *Deciphering the path to cost efficiency and sustainability in the snowflake environment*, Partners Universal International Innovation Journal. 2023, tom 1, 4, str. 231–250.

[8] Hassan, I., *Storage structures in the era of big data: From data warehouse to lakehouse*, Journal of Theoretical and Applied Information Technology. 2024, tom 102, 6.

[9] Heilig, L., Voß, S., *Managing cloud-based big data platforms: a reference architecture and cost perspective*, Big data management. 2017, str. 29–45.

[10] Immidi, S., *Error metrics used in time series forecasting modeling*, `https://medium.com/analytics-vidhya/error-metrics-used-in-time-series-forecasting-modeling-9f068bdd31ca`. 2024.

[11] Jain, P., Kraft, P., Power, C., Das, T., Stoica, I., Zaharia, M.A., *Analyzing and comparing lakehouse storage systems*, w: *Conference on Innovative Data Systems Research* (Conference on Innovative Data Systems Research, 2023).

[12] Kothari, V., *Time series evaluation metrics — mape vs wmape vs smape — which one to use, why and when? — part1*, `https://medium.com/@vinitkothari.24/time-series-evaluation-metrics-mape-vs-wmape-vs-smape-which-one-to-use-why-and-when-part1-32d3852b4779`. 2024.

[13] Kowieski, J., *What is a cloud data platform and why should you use one?*, `https://www.thoughtspot.com/data-trends/cloud/cloud-data-platform`. 2023.

[14] List, B., Bruckner, R.M., Machaczek, K., Schiefer, J., *A comparison of data warehouse development methodologies case study of the process warehouse*, w: *Database and Expert Systems Applications: 13th International Conference, DEXA 2002 Aix-en-Provence, France, September 2–6, 2002 Proceedings 13* (Springer, 2002), str. 203–215.

[15] Loghin, V., *Implementing medallion architecture in snowflake*, https://medium.com/@valentin.loghin/implementing-medallion-architecture-in-snowflake-4e1539d23c09. 2024.

[16] Medium, *Evaluation metrics for classification*, https://medium.com/@impythonprogrammer/evaluation-metrics-for-classification-fc770511052d. 2024.

[17] Medium, *Understanding log loss: A comprehensive guide with code examples*, https://medium.com/@TheDataScience-ProF/understanding-log-loss-a-comprehensive-guide-with-code-examples-c79cf5411426. 2024.

[18] Nambiar, A., Mundra, D., *An overview of data warehouse and data lake in modern enterprise data management*, Big Data and Cognitive Computing. 2022, tom 6, str. 132.

[19] Needleman, P., *The snowflake lakehouse?*, https://medium.com/snowflake/the-snowflake-lakehouse-b583746d686b. 2024.

[20] Pala, S.K., *Databricks analytics: Empowering data processing*, Machine Learning and Real-Time Analytics. Eduzone: International Peer Reviewed/Refereed Multidisciplinary Journal. 2021, tom 10, 1, str. 76–82.

[21] Pokrywka, D., *Github repository with scripts and generated data*, https://github.com/Pokrywkaa/Comaprison_Snowflake_Databricks. 2024.

[22] Schneider, J., Gröger, C., Lutsch, A., Schwarz, H., Mitschang, B., *Assessing the lakehouse: Analysis, requirements and definition.*, w: *ICEIS (1)* (2023), str. 44–56.

[23] Shan, J., *Medallion architecture: What, why and how*, https://medium.com/@junshan0/medallion-architecture-what-why-and-how-ce07421ef06f. 2024.

[24] Sharma, V., *A comprehensive guide on delta lake*, https://www.analyticsvidhya.com/blog/2023/02/a-comprehensive-guide-on-delta-lake/. 2024.

[25] Singh, A., *How: New view at catalyst optimizer in apache spark*, https://medium.com/@diehardankush/how-new-view-at-catalyst-optimizer-in-apache-spark-739cd335d145. 2023.

[26] Singh, A., Ahmad, S., *Architecture of data lake*, International Journal of Scientific Research in Computer Science Engineering and Information Technology. 2019, tom 5, str. 411–414.

[27] Snowflake, Inc., *Snowflake documentation*, https://docs.snowflake.com/. 2024.

[28] Stellwall, M., *Snowpark ml, a machine learning toolkit for snowflake*, https://medium.com/snowflake/snowpark-ml-a-machine-learning-toolkit-for-snowflake-4119b0bf204. 2024.

[29] Tufă, A., *Self-organizing data layouts for databricks delta*, CWI. 2019, tom 20, str. 29.

LIST OF FIGURES

LIST OF TABLES