**ZADANIE 2**

**Delta Lake**

**Technologia natywna dla Databricks, rozwijana przez twórców platformy.**

**Zalety:**

* Pełna integracja z ekosystemem Databricks – maksymalna wydajność i wsparcie techniczne.
* Prosta implementacja ACID, time travel oraz zarządzania schematem.
* Świetne wsparcie dla batch + streaming (Structured Streaming).
* Doskonała optymalizacja pod kątem performance (Z-Ordering, caching, data skipping).
* Bardzo dobre wsparcie narzędzi do zarządzania i monitorowania.

**Optymalne scenariusze użycia:**

* Organizacje, które pracują głównie w Databricks.
* Budowa warstwy Lakehouse – hurtownie danych w chmurze.
* Przetwarzanie danych batch i streaming.
* Audytowalność i wersjonowanie danych.
* Szybki rozwój pipeline’ów ETL z wykorzystaniem Spark.

**Apache Iceberg**

**Format otwarty, zaprojektowany z myślą o wysokiej elastyczności i interoperacyjności.**

**Zalety:**

* Obsługuje dynamiczną ewolucję partycjonowania i schematu (np. dodawanie/usuwanie partycji bez przepisania danych).
* Bardziej zaawansowane zarządzanie metadanymi – lepsza skalowalność przy dużych wolumenach danych.
* Zgodność z wieloma silnikami analitycznymi (Presto, Trino, Flink, Hive, Snowflake).
* Lepsza separacja warstwy compute i storage – idealne dla rozproszonych środowisk.

**Optymalne scenariusze użycia:**

* Organizacje z heterogenicznym środowiskiem analitycznym – nie tylko Databricks.
* Potrzeba budowy otwartej, wielosilnikowej platformy danych.
* Bardzo duże zbiory danych, wymagające zaawansowanej optymalizacji metadanych.
* Długofalowe inwestycje w open source i standaryzację formatu danych.

**ZADANIE 3**

Architektura medalionowa (Bronze, Silver, Gold) w Databricks to popularne podejście do organizacji warstw danych, jednak nie jest wolna od wad. Poniżej przedstawiono 20 krytycznych punktów, które mogą wskazywać, że medallion nie zawsze jest optymalnym rozwiązaniem.

**1. Złożoność architektoniczna:** Każda warstwa dodaje dodatkowe tabele, transformacje i zarządzanie – rośnie złożoność systemu.

**2. Wydłużony czas przetwarzania danych:** Przejście przez 3 warstwy powoduje opóźnienia – time-to-insight może być zbyt długi.

**3. Nadmiarowość danych:** Dane są kopiowane z warstwy do warstwy, co zwiększa zużycie przestrzeni dyskowej.

**4. Trudność w śledzeniu pochodzenia danych (data lineage):** Więcej warstw = więcej miejsc, które trzeba uwzględnić w analizie pochodzenia danych.

**5. Skalowanie zespołów utrudnione:** Zespół musi rozumieć i zarządzać wszystkimi warstwami – bariera wejścia dla nowych członków.

**6. Niejasne granice między warstwami:** Różnica między Silver a Gold może być rozmyta – trudność w definiowaniu reguł przejścia.

**7. Zduplikowana logika transformacji:** Niektóre operacje są powielane między warstwami (np. czyszczenie danych).

**8. Nadmierne zależności między pipeline'ami:** Każda warstwa zależy od poprzedniej – jeden błąd propaguje się dalej.

**9. Trudność w iteracyjnym rozwoju:** Zmiana logiki w Bronze wymaga aktualizacji Silver i Gold – spowalnia wdrażanie zmian.

**10. Koszty operacyjne:** Więcej warstw = więcej harmonogramów, jobów, notebooków = wyższy koszt utrzymania.

**11. Niskie ROI przy prostych przypadkach użycia:** Dla małych firm lub prostych analiz – medallion to 'overkill'.

**12. Utrudnione testowanie i debugowanie:** Trzeba śledzić błędy przez wszystkie warstwy, zamiast w jednym miejscu.

**13. Nadmierna ilość tabel:** Każda warstwa generuje nowe tabele – łatwo wpaść w chaos nazw i struktur.

**14. Potrzeba wysokiej dojrzałości zespołu danych:** Architektura ta zakłada doświadczony zespół – nie nadaje się dla początkujących.

**15. Trudność z utrzymaniem spójności danych:** Różne wersje danych w różnych warstwach mogą prowadzić do niespójności.

**16. Problemy z aktualizacjami danych historycznych:** Zmiana danych w Bronze wymaga aktualizacji całego łańcucha.

**17. Nadmierna liczba checkpointów i retry logic:** Każda warstwa wymaga własnego monitorowania i retry – więcej pułapek błędów.

**18. Brak elastyczności w ad hocowych analizach:** Dane często są zbyt przetworzone w Gold – trudniej zrobić analizę niestandardową.

**19. Powolny onboarding danych z zewnętrznych źródeł:** Nowe dane muszą przejść przez każdą warstwę zanim będą dostępne dla analityków.

**20. Architektura 'dla architektury':** Często wdrażana bez realnej potrzeby – bo 'tak robią wszyscy', nie dlatego, że jest to optymalne.