Zadanie 1 – Szacunkowy roczny koszt projektu Lakehouse na Azure

Założenia:

- Start: 10 TB historycznych danych
- Przyrost: 200 GB/miesięcznie → +2,4 TB rocznie → końcowy rozmiar ~12,4 TB
- Cały czas przechowywane w magazynie (hot tier GPv2)

Komponenty i ceny (przybliżone ceny PAYG w USD; kurs PLN zależy od faktora przeliczeniowego):

1. **Azure Storage Account (Blob/ADLS Gen2)**
    - 12,4 TB × 730 GB = ~9 060 GB-mies.
    - *Hot tier:* ~0,020 USD/GB-mies. → 9 060 × 0,020 = **181 USD/mies.** → **2 172 USD/rok**
2. **Azure Databricks**
    - Przyjmijmy plan płatności PAYG: np. 1 węzeł premium DBU ~0,40 USD/DBU. Przy 8 DBU ciągle pracujące →
    - 8 × 0,4 = 3,2 USD/h → ~2 304 USD/mies. (~27 648 USD/rok)
3. **Azure Key Vault**
    - Standard: 1 USD/klucz/mies. + operacje: 0,03 USD/10 000 operacji. Załóżmy 5 kluczy + 1 mln operacji → 5 USD + 3 USD = **8 USD/mies.** → **96 USD/rok**
4. **Azure Data Factory**
    - Szacujemy 100 pipeline runs/mies. → ~1 USD za pipeline-run → **100 USD/mies.** → **1 200 USD/rok**
5. **SQL Server (np. Managed Instance)**
    - Załóżmy 4 vCore Business Critical: ok. 1 200 USD/mies. → **14 400 USD/rok**
6. **Transfer danych (wewnętrzny/wyjściowy)**
    - Zakładamy 5 TB/rok dane wychodzące poza region → 5 000 GB × 0,087 = **435 USD/rok**

Zadanie 2 – Porównanie: Delta Lake vs Apache Iceberg

Delta Lake

- Opracowany przez Databricks, domyślnie wspierany w ekosystemie Spark.
- Idealny dla środowisk opartych o Databricks i Azure.

Apache Iceberg

- Open-source'owa technologia zaprojektowana z myślą o dużej skalowalności i interoperacyjności.
- Doskonała do pracy z wieloma silnikami: Spark, Flink, Trino, Presto, Hive.

Główne różnice technologiczne

ACID transakcje

- **Delta Lake:** log transakcyjny (parquet + JSON); dobre wsparcie w Spark.
- **Iceberg:** pełne MVCC i snapshoty; bardziej odporne na błędy w środowiskach rozproszonych.

Ewolucja schematu

- **Delta Lake:** obsługuje mergeSchema i overwriteSchema.
- **Iceberg:** potężna kontrola wersji – możliwość branchowania, rewizji, rollbacków.

Optymalizacja zapytań

- **Delta Lake:** Z-order clustering, data skipping.
- **Iceberg:** Zaawansowany partitioning spec, sortowanie, plany czytania inkrementalnego.

Obsługa danych historycznych

- **Delta Lake:** Time travel do checkpointa (retention).
- **Iceberg:** Snapshoty z metadanymi; możliwość porównań między wersjami.

Kompatybilność silników

- **Delta Lake:** Spark-centric.
- **Iceberg:** Spark, Flink, Trino, Hive – szerokie zastosowanie w środowiskach heterogenicznych.

Zarządzanie metadanymi

- **Delta Lake:** metadane w logach JSON.
- **Iceberg:** specjalna struktura plików metadanych z manifestami – szybsze i skalowalne.

Rekomendacje zastosowań

Kiedy wybrać Delta Lake?

- Jesteś w ekosystemie Azure i Databricks.
- Operujesz głównie w Spark.
- Potrzebujesz prostoty i gotowych optymalizacji.
- Projekty mają umiarkowaną złożoność i skalę.

Kiedy wybrać Iceberg?

- Masz wielosilnikowe środowisko (Flink, Hive, Presto...).
- Duża skala danych i potrzeba silnego versioningu.
- Potrzebujesz funkcji typu branching, rollback, audit trail.
- Chcesz uniknąć vendor lock-in.

Zadanie 3 – Krytyka architektury medali w Lakehouse

Poniżej 20 punktów krytycznego spojrzenia na architekturę medallion (bronze-silver-gold):

1. **Złożoność operacyjna** – wymaga zarządzania wieloma strefami i pipeline'ami.
2. **Koszty przetwarzania** – każda strefa generuje obciążenie i koszt.
3. **Utrzymanie spójności** – niezbędny mechanizm walidacji danych między strefami.
4. **Opóźnienia** – dane mogą być opóźnione przez sequential execution.
5. **Redundantna przechowywalnia** – wielokrotne kopie danych.
6. **Wyzwania z wersjonowaniem** – synchronizacja wielu snapshotów.
7. **Złożoność ryzyka** – błędy w strefie niższej propagują się wyżej.
8. **Skalowalność** – większe zasoby CPU/RAM potrzebne z czasem.
9. **Koszty operacyjne** – więcej pipeline'ów = więcej DevOps.
10. **Nadmierna generalizacja** – nie zawsze potrzebne gold-strefy.
11. **Trudności z ewolucją** – rewalidacja przy zmianie wymagań.
12. **Skupienie na strukturze** – może zniechęcać eksplorację ad-hoc.
13. **Monitorowanie** – każda warstwa wymaga osobnych alertów i checków.
14. **Obciążenie sieci** – kopiowanie bloków między strefami.
15. **Zarządzanie kosztami licencji** – DBU, storage, compute mnożą się.
16. **Możliwość over-engineeringu** – proste przypadki nie potrzebują pełnego stacku.
17. **Ryzyko opóźnień przy zmianach** – zmiana w bronze wymusza przebudowę silver/gold.
18. **Testowanie regresji** – każda strefa wymaga własnych testów.
19. **Kultura danych** – wymaga uzgodnionych standardów i współpracy.
20. **Utrudniona ad-hoc analiza** – żeby dostać dane gold trzeba przejść przez cały pipeline.