Pertemuan 8 — UTS: Tugas Besar

"*Pipeline* ETL bronze-silver-gold", Capaian A: *pipeline* ETL di satu mesin

---

Hendri Karisma

Semester Ganjil 2026/2027

Program Studi Teknik Informatika, STMIK Tazkia

Alur pertemuan hari ini

1. Tujuan dan kaitan dengan CPMK
2. Pilihan *dataset* dan skenario
3. Arsitektur acuan
4. Kebutuhan wajib dan nilai lebih
5. Kebijakan *tiering* dan perhitungan *cost*
6. Protokol *data quality test* dan *idempotency*
7. Repositori, laporan, dan demo
8. Rubrik penilaian
9. Aturan main, jadwal, dan *checklist*
10. Pertanyaan yang akan muncul saat demo

Bobot dan cakupan

UTS bernilai **35%** nilai akhir, mengukur CPMK-1, CPMK-2, dan CPMK-3, dan mencakup seluruh materi pertemuan 1–7: *data modeling* dan format, *storage* dan *tiering*, *lakehouse* dan medallion, ETL satu mesin (Arrow, Polars, DuckDB), *orchestration*, *data quality*, dan *serving store*. Satu mesin; Spark **tidak** wajib (Spark adalah inti UAS).

Tujuan

Mengapa UTS berbentuk *pipeline*, bukan ujian tertulis

Di industri, *data engineer* tidak dinilai dari hafalan definisi *lakehouse*, tetapi dari pertanyaan sederhana:

- Apakah datanya sampai, benar, dan tepat waktu?
- Apa yang terjadi bila *pipeline* dijalankan ulang?
- Berapa *cost*-nya tahun depan, ketika datanya sepuluh kali lipat?

Yang Anda hasilkan

Satu *pipeline* ETL utuh di satu mesin yang menyala dengan satu perintah: *raw data* masuk, *layer* bronze–silver–gold terbentuk, *data quality test* menjaga gerbangnya, *orchestrator* menjalankannya setiap hari, dan satu analisis bisnis berdiri di atasnya.

Ringkasan

Sistem yang berjalan dan dapat dijelaskan lebih berharga daripada sistem besar yang tidak dipahami pembuatnya.

Kaitan dengan CPMK

CPMK	Rumusan singkat	Dibuktikan oleh
CPMK-1	Peran *data engineering*, pemodelan OLTP/OLAP, pemilihan format	*Star schema gold layer*, *grain* yang dinyatakan, alasan Parquet dan *partition*
CPMK-2	*Storage layer*, *tiering*, *lakehouse* dan medallion	*Object storage* berpartisi, tabel *lakehouse* bronze/silver/gold dengan aturan tiap *layer*, kebijakan *tiering* dan *cost*-nya
CPMK-3	ETL satu mesin, *orchestration*, *idempotent*, teruji; memilih *serving store*	*Transformation* Polars/DuckDB/dbt, DAG harian, *backfill* yang terbukti *idempotent*, *data quality test* yang menghentikan *pipeline*, alasan *serving store* untuk gold

Dataset dan skenario

Dua jalur *dataset*

Jalur A — bank syariah digital (sintetis)

Pembangkit data dibagikan di kelas: nasabah, rekening, akad pembiayaan, transaksi harian, cabang.

- Skala dapat diatur: mulai 1 juta, disarankan $\geq$ 10 juta baris transaksi
- Mengandung "kotoran" yang disengaja: duplikat, nilai kosong, data terlambat, kode akad tidak valid
- Perubahan profil nasabah untuk SCD tipe 2

Jalur B — data terbuka

Dataset nyata dari portal data terbuka pemerintah (misalnya Satu Data Indonesia atau portal data provinsi) atau sumber terbuka lain dengan lisensi yang jelas.

- Harus memiliki dimensi waktu dan bisa diperbarui berkala (atau disimulasikan per hari)
- Minimal dua sumber yang digabung (misalnya data transportasi + data wilayah)
- Disetujui dosen saat penilaian rancangan (P6, 9 November)

Larangan

Tidak boleh memakai data pribadi sungguhan dari tempat kerja, magang, atau hasil *scraping* yang memuat identitas orang. Pelanggaran membuat tugas bernilai nol.

Skenario contoh (jalur A)

Direksi bank syariah digital ingin *dashboard* pagi yang menjawab:

1. Nominal dan jumlah transaksi kemarin per cabang dan per kanal
2. Pembiayaan cair per jenis akad, dan tren 30 hari
3. Nasabah baru yang belum bertransaksi dalam 14 hari
4. Cabang dengan lonjakan transaksi tidak wajar dibanding rata-rata 4 pekan

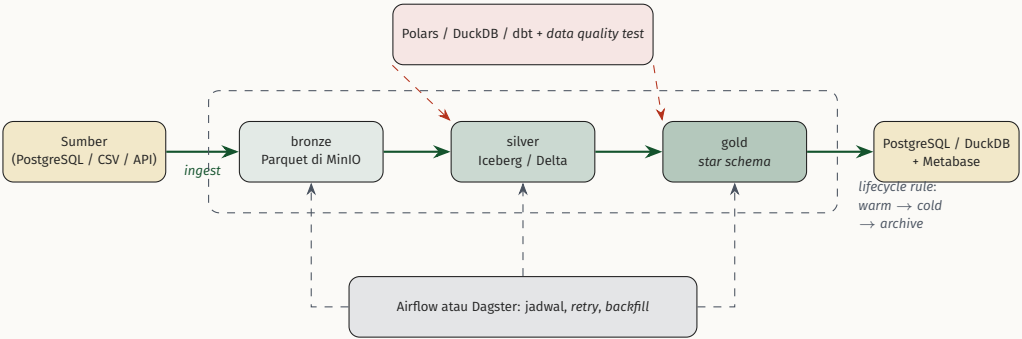
Data harus siap sebelum pukul 06.00, dan data dua tahun terakhir harus tetap dapat di-*query* untuk analisis tahunan.

Terjemahan teknis

- *Grain* fakta: satu transaksi
- Dimensi: tanggal, cabang, kanal, nasabah (SCD-2), akad
- SLA: *partition* H-1 lengkap sebelum 06.00
- Retensi *warm* 13 bulan, *cold* sampai 2 tahun, *archive* sesudahnya
- *Tokenization* PII (NIK, nomor HP) di silver
- *Data quality test* di gerbang silver dan gold

Arsitektur acuan

Arsitektur acuan



Semua komponen berjalan di laptop dengan `docker compose up`. Komponen boleh diganti dengan padanannya selama alasannya ditulis di laporan.

Pilihan *tool* dan *when to use*

Layer	Pilihan	Kapan memilih
Transformation	dbt-duckdb (SQL) atau Polars (Python)	dbt bila *transformation* relasional dan ingin *test* dan *lineage* bawaan; Polars bila logika lebih nyaman di Python. Spark tidak dipakai di UTS
Query engine	DuckDB, Polars, DataFusion	Semuanya satu mesin; Trino opsional bila ingin katalog bersama
Format tabel	Iceberg (PyIceberg) atau Delta (delta-rs)	Iceberg bila ingin netral *engine*; Delta bila nyaman dengan delta-rs
Orchestrator	Airflow atau Dagster	Airflow bila ingin standar industri; Dagster bila ingin berpikir per *asset* dan menguji lokal
Serving	PostgreSQL atau DuckDB + Metabase-/Superset	PostgreSQL bila banyak pengguna *dashboard*; DuckDB untuk demo lokal (P7)

Ringkasan

Nilai tidak ditentukan oleh banyaknya alat. *Stack* kecil yang dipilih dengan alasan dan berjalan stabil dinilai lebih tinggi daripada *stack* besar yang setengah jadi.

Kebutuhan wajib

Kebutuhan wajib (W1–W10)

W1 *Ingest* ke bronze dalam Parquet, berpartisi tanggal, dengan kolom metadata (`_ingested_at`, `_source`, `_batch_id`).

W2 *Layer* silver: tipe data benar, duplikat dibuang, kode distandardkan, PII di-*tokenize*.

W3 *Gold layer*: *star schema* dengan *grain* yang dinyatakan; minimal satu fakta dan tiga dimensi.

W4 Satu dimensi memakai SCD tipe 2.

W5 Tabel silver atau gold memakai format *lakehouse* (Iceberg atau Delta) dengan bukti *time travel*.

W6 DAG harian di Airflow atau Dagster dengan *retry*.

W7 Seluruh tugas *idempotent*; *backfill* 7 hari terbukti tidak menggandakan data.

W8 Minimal 8 *data quality test* yang mencakup 5 dimensi; uji gagal menghentikan pembaruan gold.

W9 Kebijakan *tiering* tertulis, *lifecycle rule* terpasang di *object storage*, dan perhitungan *cost* 3 tahun.

W10 Gold dimuat ke *serving store*; satu *dashboard* atau *query* analitik menjawab minimal tiga pertanyaan bisnis.

Seluruh sistem menyala dengan satu perintah dan dijalankan ulang dari nol saat demo.

Nilai lebih (tidak wajib, maksimal +10%)

- Perbandingan terukur pandas vs Polars vs DuckDB vs DataFusion pada *transformation* yang sama (waktu, memori)
- *Transformation* ditulis dua kali (dbt dan Polars) dengan hasil identik
- Trino dengan katalog Iceberg dipakai bersama oleh dua *engine*
- CI di GitHub/GitLab yang menjalankan `dbt build` pada data contoh
- Pemantauan *freshness* dan volume dengan *alert*
- ClickHouse sebagai *serving store* dengan pengukuran pola akses (P7)
- Jejak audit gold dengan *hash chain* atau immudb (P7)

Ringkasan

Nilai lebih hanya dihitung bila seluruh W1–W10 sudah berjalan. *Feature* tambahan di atas fondasi yang rapuh tidak menambah nilai.

Tiering dan *cost*

Kebijakan *tiering*: bentuk yang diminta

Data	*Hot / warm*	*Cold*	*Archive*	Dihapus
bronze	0–30 hari	31 hari–1 tahun	1–5 tahun	sesudah retensi
silver	0–13 bulan	13–24 bulan	…	…
gold	selama dipakai *dashboard*	…	…	…

(angka di atas contoh pengisian, bukan jawaban)

Setiap baris wajib beralasan

Siapa yang membaca data ini, seberapa sering, seberapa cepat harus kembali, dan aturan apa yang menentukan retensinya (kebutuhan audit, kebijakan internal, regulasi).

Bukti di sistem

Lifecycle rule MinIO (atau padanannya) terpasang dan terlihat di *configuration* repositori. MinIO lokal tidak benar-benar memindahkan ke Glacier; cukup tunjukkan aturan dan simulasinya.

Perhitungan *cost* 3 tahun

Model sederhana per bulan m:

$$B_m = \sum_{t \in tier} V_{t,m}\, h_t \;+\; R_m\, h_{\text{ambil}} \;+\; E_m\, h_{egress}$$

- $V_{t,m}$: volume (GB) di *tier* t pada bulan m, dari laju data harian dan kebijakan Anda
- $\sum_t$ (dibaca "sigma"): jumlahkan untuk semua *tier* t
- h_t: harga simpan per GB per bulan
- R_m, E_m: volume yang diambil dari *cold*/*archive* dan yang keluar jaringan

Bandingkan tiga skenario: semua *hot*, kebijakan Anda, dan kebijakan paling hemat. Tunjukkan titik ketika kebijakan Anda mulai menguntungkan.

Harga

Ambil harga dari halaman resmi penyedia (S3, GCS, Azure, atau penyedia lokal) pada minggu pengerjaan, cantumkan tanggal aksesnya. Harga berubah; yang dinilai adalah **model dan alasannya**, bukan angka mutlaknya.

Asumsi pertumbuhan

Nyatakan laju data per hari dan pertumbuhannya (misalnya naik 5% per bulan), lalu tunjukkan sensitivitas: apa yang berubah bila laju dua kali lipat?

Protokol *data quality test* dan *idempotency*

Tiga uji yang dijalankan saat demo

Uji	Yang dilakukan	Hasil yang diharapkan
Q1	Jalankan DAG untuk satu hari dua kali berturut-turut	Jumlah baris dan total nominal di silver dan gold identik setelah eksekusi pertama dan kedua
Q2	Dosen menyisipkan data rusak (duplikat, nilai di luar domain, kolom wajib kosong) ke sumber	*Data quality test* gagal, DAG berhenti, gold tidak berubah, *error message* menunjuk baris dan aturan yang dilanggar
Q3	*Backfill* 7 hari, lalu ulangi *backfill* yang sama	Hasil identik; waktu eksekusi dicatat; tidak ada *partition* yang hilang atau ganda

Catat sebelum menjalankan

Tulis dugaan hasil tiap uji di laporan sebelum menjalankannya, lalu bandingkan dengan hasil sebenarnya. Dugaan yang meleset tetapi dijelaskan dengan baik tetap bernilai.

Bukti

Query pemeriksaan disimpan di repositori (`checks/`), sehingga siapa pun dapat mengulang dan mendapat angka yang sama.

Contoh *query* pemeriksaan

```sql
-- checks/q1_idempoten.sql
select tgl,
       count(*)        as baris,
       sum(nominal)    as total,
       count(distinct trx_id) as trx_unik
from gold.fct_transaksi
where tgl between date '2026-11-01'
              and date '2026-11-07'
group by tgl
order by tgl;
```

tgl	baris (1)	baris (2)
2026-11-01	…	…
2026-11-02	…	…
…		

Tabel di laporan memuat hasil eksekusi pertama dan kedua berdampingan, plus kolom selisih yang harus nol. `baris` sama dengan `trx_unik` membuktikan tidak ada duplikat.

Repositori, laporan, dan demo

Struktur repositori

```
uts-de-kelompok-X/
  README.md          # cara menyalakan
  docker-compose.yml
  .env.example       # tanpa rahasia asli
  generator/         # jalur A / unduh B
  ingest/
  dags/              # Airflow atau Dagster
  dbt/  atau  transform/  # Polars
  checks/            # kueri Q1-Q3
  infra/minio-lifecycle.json
  biaya/biaya_3_tahun.ipynb
  laporan/laporan.pdf
```

```yaml
# docker-compose.yml (kerangka minimal)
services:
  minio:
    image: minio/minio
    command: server /data
  postgres:
    image: postgres:16
  airflow:
    image: apache/airflow
    command: standalone
    volumes: ["./dags:/opt/airflow/dags",
              "./dbt:/opt/airflow/dbt"]
  metabase:
    image: metabase/metabase
```

Versi *image* disematkan (*pinned*) di *file* akhir agar sistem tetap menyala bulan depan.

Struktur laporan (maksimal 12 halaman)

1. Ringkasan: masalah, solusi, hasil utama (setengah halaman)
2. *Dataset* dan pertanyaan bisnis
3. Arsitektur dan alasan tiap pilihan alat
4. Pemodelan: *grain*, *star schema*, SCD
5. *Bronze layer*, silver, gold dan format tabel
6. *Orchestration*, *idempotency*, *backfill* (Q1, Q3)
7. *Data quality*: daftar uji, hasil Q2
8. *Tiering* dan *cost* 3 tahun, dengan sensitivitas
9. Analisis bisnis dari *gold layer*
10. Keterbatasan, pembagian kerja, penggunaan alat bantu AI

Ringkasan

Yang dinilai adalah analisis. “Kami memakai Iceberg” tidak bernilai; “kami memakai Iceberg karena …, dan *cost*-nya …” bernilai.

Presentasi dan demo (15 menit per kelompok)

1. 3 menit: masalah, arsitektur, pilihan alat
2. 7 menit: demo langsung
   - `docker compose up` dari kondisi bersih
   - DAG berjalan, *layer* terisi
   - Uji Q1–Q3
   - *Dashboard* menjawab pertanyaan bisnis
3. 5 menit: tanya jawab, semua anggota menjawab

Rencana cadangan

Siapkan rekaman layar demo lengkap. Bila demo langsung gagal karena hal di luar kendali (listrik, jaringan), rekaman dipakai dan dosen mengajukan pertanyaan tambahan tentang bagian yang gagal.

Laptop presentasi

Unduh semua *image* sebelum hari presentasi. Jangan bergantung pada internet kampus.

Penilaian dan aturan main

Rubrik penilaian

Komponen	Bobot	Nilai penuh bila
Arsitektur dan pemodelan	15%	Diagram jelas, alasan tiap alat, *grain* dan *star schema* tepat, SCD-2 benar (W3, W4)
Implementasi *pipeline*	20%	W1, W2, W5 berjalan; satu perintah menyalakan sistem; kode rapi
Orchestration dan *idempotency*	15%	W6, W7; Q1 dan Q3 lulus dengan bukti *query*
Data quality	10%	W8; Q2 menghentikan *pipeline* dengan *error message* yang jelas
Tiering dan *cost*	10%	W9; model *cost* beralasan, sumber harga bertanggal, sensitivitas
Analisis di *gold layer*	10%	W10 menjawab pertanyaan bisnis dengan angka yang benar
Laporan	10%	Lengkap, analitis, keterbatasan jujur
Presentasi dan demo	10%	Demo berhasil; semua anggota menjawab dengan paham
Nilai lebih	*+10%*	*Perbandingan engine satu mesin, Trino, CI, observability, ClickHouse, audit trail immutable*

Nilai akhir UTS dibatasi 100. UTS bernilai 35% dari nilai mata kuliah.

Aturan main

- Kelompok 3–4 orang; pembagian kerja ditulis dan tercermin di riwayat Git.
- Setiap anggota harus mampu menjelaskan bagian mana pun dari sistem; nilai individu dapat dikurangi bila tidak mampu.
- Alat bantu AI boleh dipakai; tuliskan bagian mana yang dibantu. Kode yang tidak dapat dijelaskan dianggap bukan karya kelompok.
- Menyalin repositori kelompok lain atau sumber daring tanpa menyebut sumber dinilai nol.
- Data pribadi **selalu sintetis** (jalur A) atau data terbuka tanpa identitas orang (jalur B).
- Tidak ada kata sandi, *key* akses, atau token di repositori. Pakai `.env` yang tidak di-*commit* dan sediakan `.env.example`.
- Keterlambatan pengumpulan: dikurangi 10 poin per hari, maksimal 3 hari.
- Repositori dibekukan dengan *tag* `uts-final`; perubahan sesudah tenggat tidak dinilai.

Jadwal



Rancangan (minggu 6) dan progres (minggu 7) dinilai sebagai tugas dan masuk komponen “tugas, praktikum, dan kuis” (20%), bukan nilai UTS. Tanggal mengikuti jadwal Senin; bila kalender akademik bergeser, tanggal menyesuaikan.

Checklist pengumpulan

- ☐ `docker compose up` dari kondisi bersih berhasil di laptop lain
- ☐ README menjelaskan cara menyalakan dan menjalankan DAG
- ☐ W1–W10 dapat ditunjukkan satu per satu
- ☐ *Query* Q1–Q3 ada di `checks/` beserta hasilnya
- ☐ *Lifecycle rule* ada di `infra/`
- ☐ Perhitungan *cost* dengan tanggal akses harga
- ☐ Laporan $\leq$ 12 halaman, PDF
- ☐ Rekaman demo cadangan
- ☐ Tidak ada rahasia di riwayat Git (periksa dengan `git log -p`)
- ☐ *Tag* `uts-final` dibuat sebelum tenggat

Pertanyaan yang akan muncul saat demo

1. Apa *grain* tabel fakta Anda, dan mengapa bukan yang lain?
2. Tunjukkan baris nasabah yang berubah alamat di dimensi SCD-2.
3. Mengapa *partition* ini per hari, bukan per bulan?
4. Tunjukkan *query plan query dashboard* terlama. Bagian mana yang mahal?
5. Mengapa Polars/DuckDB cukup untuk data sebesar ini, dan pada ukuran berapa Anda pindah ke Spark?
6. Bila *task* ingest gagal di tengah jalan lalu di-*retry*, apa yang mencegah data ganda?
7. *Data quality test* mana yang paling sering gagal, dan apa artinya bagi sumber data?
8. Data bronze berumur dua tahun ada di *tier* mana? Berapa *cost* dan waktu untuk mengambilnya kembali?
9. Bila data tumbuh sepuluh kali, komponen mana yang pertama menjadi hambatan?
10. Apa satu hal yang akan Anda ubah bila mengulang dari awal?

Setelah UTS

Paruh pertama membangun *pipeline* ETL bronze-silver-gold di **satu mesin**: murah, dapat diulang, dan cukup untuk data puluhan GB.

Paruh kedua membawa *pipeline* yang sama ke **skala besar** dengan Apache Spark di *cluster* (HPC data), lalu menambah *hot path*, *dashboard* atau *data app*, *security*, dan *governance*. Hasil UTS Anda menjadi *baseline* satu mesin yang dibandingkan dengan Spark di UAS.

Pertemuan 9 (Senin, 30 November)

Dari satu mesin ke *cluster*: ekosistem Hadoop dan Apache Spark dari dalam (RDD, DAG, *lineage*), serta *job* Spark dalam Scala dan PySpark.

Selamat mengerjakan

Pertanyaan?