Pertemuan 13 — *Data-driven*: dari *Metric* ke *Dashboard* dan *Data App*

KPI, *semantic layer*, Superset/Metabase, Streamlit, dan eksperimen A/B

Hendri Karisma

Semester Ganjil 2026/2027

Program Studi Teknik Informatika, STMIK Tazkia

Alur pertemuan hari ini

1. Dari pertanyaan bisnis ke *metric*
2. *North star*, KPI, *guardrail*
3. Satu definisi untuk satu *metric*: *semantic layer*
4. Empat jenis analitik dan analitik di atas *pipeline*
5. Titik temu dengan *Machine Learning*
6. Eksperimen A/B yang dapat dipercaya
7. *Data product*, *data mesh*, dan bias
8. Membangun *dashboard* dengan Superset atau Metabase
9. Visualisasi dan *storytelling*
10. Membangun *data app* dengan Streamlit atau FastAPI
11. Praktikum dan persiapan demo UAS

Sasaran pertemuan ini

Anda dapat menurunkan pertanyaan bisnis menjadi *metric* yang terdefinisi jelas, lalu menyajikannya di atas *layer gold* sebagai *dashboard* atau *data app* yang benar-benar dipakai untuk mengambil keputusan.

Dari pertanyaan bisnis ke *metric*

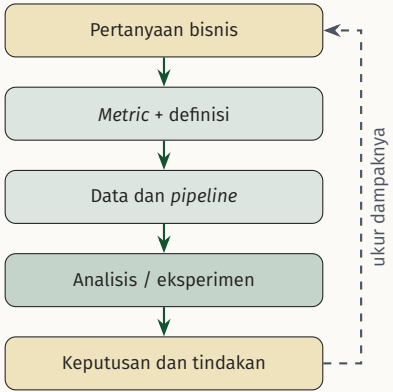
Pipeline yang sempurna pun bisa tidak berguna

Dua belas pertemuan terakhir kita membangun mesin: *lakehouse*, *streaming*, *hot* dan *cold*. Hari ini pertanyaannya bergeser:

- Keputusan apa yang berubah karena data ini?
- Siapa yang mengambil keputusan itu?
- Bagaimana kita tahu keputusannya lebih baik?

Gejala platform tanpa arah

Ratusan tabel *gold*, puluhan *dashboard*, dan tidak ada yang bisa menjawab “apakah nasabah kita bertambah sehat bulan ini?”



Menurunkan pertanyaan menjadi *metric*

Pertanyaan bisnis	*Metric*	Definisi yang harus disepakati
Apakah nasabah baru benar-benar memakai rekeningnya?	Tingkat aktivasi 30 hari	"aktif" = minimal satu transaksi finansial buatan nasabah (bukan biaya admin) dalam 30 hari sejak pembukaan
Apakah pembiayaan kita sehat?	NPF (*Non Performing Financing*)	porsi pembiayaan dengan kolektibilitas 3–5 terhadap total pembiayaan, pada tanggal posisi tertentu
Seberapa cepat kita menghentikan penipuan?	Waktu deteksi *fraud* (median, P95)	selisih waktu transaksi dan waktu ditandai; hanya kasus terkonfirmasi
Apakah *feature* QRIS baru disukai?	Transaksi QRIS per nasabah aktif per pekan	pembilang dan penyebut dari tabel *gold* yang sama

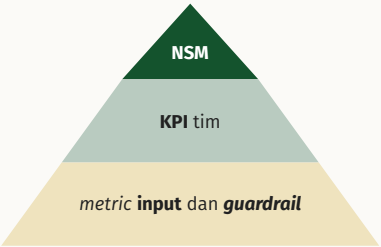
Ringkasan

Metric tanpa definisi tertulis dihitung berbeda oleh setiap tim. Kolom kanan itulah pekerjaan *analytics engineer*.

Definisi resmi kolektibilitas dan NPF mengikuti regulasi OJK; versi di atas disederhanakan untuk latihan.

North star, KPI, *guardrail*

Tiga *metric layer*



- ***North star metric*** (NSM): satu angka yang paling mewakili nilai yang diterima nasabah dan keberlanjutan bisnis.
- **KPI**: *metric* yang dipegang tiap tim dan diyakini mendorong NSM.
- ***Metric* input**: hal yang bisa langsung diubah tim (misalnya waktu pembukaan rekening).
- ***Guardrail***: *metric* yang **tidak boleh memburuk** ketika mengejar KPI (misalnya keluhan, *fraud*, NPF).

Contoh untuk bank syariah digital (ilustrasi)

NSM: jumlah nasabah aktif yang bertransaksi rutin tiap bulan. KPI tim akuisisi: aktivasi 30 hari. KPI tim pembiayaan: pembiayaan cair dengan NPF terjaga. *Guardrail*: rasio keluhan, kerugian *fraud*, NPF.

Metric yang baik, *metric* yang menyesatkan

Ciri *metric* yang baik

- Terkait langsung dengan nilai bagi nasabah atau bisnis
- Dapat dipengaruhi oleh tindakan tim
- Berupa rasio atau laju; total kumulatif yang selalu naik jarang membantu keputusan
- Definisinya tertulis dan dapat dihitung ulang dari tabel *gold*
- Ada *guardrail* yang menjaganya

Vanity metric dan hukum Goodhart

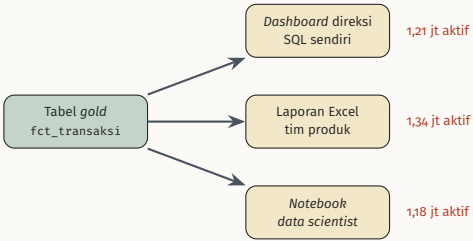
"Jumlah unduhan aplikasi" terus naik walau nasabah tidak pernah bertransaksi. Begitu sebuah ukuran dijadikan target, orang cenderung mengejar angkanya, bukan tujuannya, misalnya membuka rekening kosong demi target akuisisi.

Pertanyaan diskusi

Tim penagihan diberi KPI "NPF turun". Cara curang apa yang mungkin terjadi, dan *guardrail* apa yang mencegahnya?

Semantic layer

Masalah “dua *dashboard*, dua angka”



Angka ilustrasi.

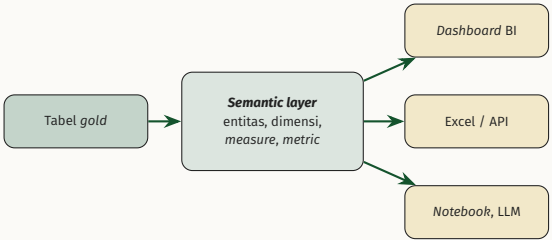
Ketiganya membaca tabel yang sama, tetapi:

- satu menghitung biaya admin sebagai transaksi,
- satu memakai 30 hari kalender, satu 30 hari bergulir,
- satu memakai zona waktu UTC, dua lainnya WIB.

Ringkasan

Rapat lalu habis untuk memperdebatkan angka mana yang benar, dan tindakannya tidak sempat diputuskan.

Semantic layer: definisi sekali, dipakai di mana-mana



- *Metric* didefinisikan sebagai kode (YAML), disimpan di Git, ditinjau seperti kode lain.
- Alat pemakai meminta ***metric* dan dimensi**, bukan menulis SQL sendiri; *semantic layer* yang menyusun SQL-nya.
- Contoh alat: **dbt Semantic Layer** (MetricFlow), **Cube**; banyak alat BI juga punya *layer* model sendiri.
- Bonus: dokumentasi dan *metric lineage* sekaligus (pertemuan 15).

Kaitan dengan *hot* dan *cold*

Definisi yang sama idealnya dipakai untuk angka sementara di ClickHouse dan angka final di *gold* (pertemuan 12). Bila belum bisa, setidaknya definisinya ditulis di satu tempat yang dirujuk keduanya.

Contoh: *metric* aktivasi di dbt *Semantic Layer*

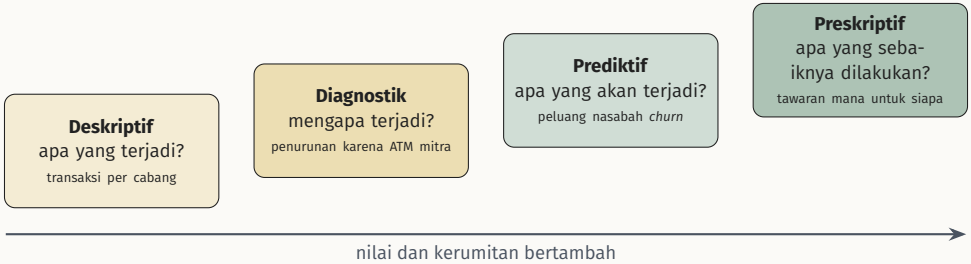
```yaml
semantic_models:
  - name: nasabah
    model: ref('dim_nasabah')
    defaults: {agg_time_dimension: tgl_buka}
    entities:
      - {name: nasabah_id, type: primary}
    dimensions:
      - {name: tgl_buka, type: time,
         type_params: {time_granularity: day}}
      - {name: segmen, type: categorical}
    measures:
      - {name: nasabah_baru, agg: count,
         expr: nasabah_id}
      - {name: aktif_30h, agg: sum,
         expr: "case when aktif_30_hari
                then 1 else 0 end"}
```

```yaml
metrics:
  - name: tingkat_aktivasi_30h
    label: Tingkat aktivasi 30 hari
    description: >
      Porsi nasabah baru dengan minimal
      satu transaksi finansial buatan
      nasabah dalam 30 hari sejak buka.
    type: ratio
    type_params:
      numerator: aktif_30h
      denominator: nasabah_baru
```

Kolom `aktif_30_hari` dihitung sekali di model dbt `dim_nasabah`. Sintaks dapat berbeda antarversi dbt; ikuti dokumentasi versi yang dipakai.

Empat jenis analitik

Dari “apa yang terjadi” ke “apa yang sebaiknya dilakukan”



- Sebagian besar nilai di organisasi masih datang dari analitik deskriptif dan diagnostik yang **benar dan tepat waktu**.
- Analitik prediktif dan preskriptif membutuhkan fondasi yang sama, ditambah label dan eksperimen.

Sudut pandang Baesens (KU Leuven)

Analitik bernilai bila menyatu dengan proses bisnis: model *churn* tidak berguna bila tim retensi tidak menerima daftarnya tepat waktu dalam bentuk yang bisa ditindaklanjuti.

Sumber: Baesens, *Analytics in a Big Data World*, Wiley, 2014.

Analitik di atas *pipeline*

Tiga kasus analitik di bank syariah digital

Churn nasabah

Tanya: siapa yang akan berhenti memakai rekening?
Data: tren transaksi, saldo, keluhan, login aplikasi.
Jalur: *cold* (skor mingguan) → *reverse ETL* ke CRM.
Ukur: retensi kelompok yang ditangani vs tidak.

Anomali dan *fraud*

Tanya: transaksi mana yang tidak wajar?
Data: *feature window* pendek, perangkat, lokasi, jaringan rekening.
Jalur: *hot* (aturan + model) dan *cold* (melatih ulang).
Ukur: *precision*, *recall*, waktu deteksi, kerugian.

Credit scoring

Tanya: layakkah pembiayaan murabahah ini?
Data: riwayat pembayaran, arus kas rekening, data biro kredit.
Jalur: *cold* untuk melatih, *online* saat pengajuan.
Ukur: NPF per angkatan pembiayaan.

Kewajiban yang ikut melekat

Keputusan yang berdampak pada nasabah (menolak pembiayaan, memblokir transaksi) harus dapat dijelaskan dan diaudit, memakai data dengan dasar pemrosesan yang sah (UU PDP, pertemuan 15), dan diperiksa biasnya.

Titik temu dengan *Machine Learning*

Feature, *label*, dan *training-serving skew*

- ***Feature***: *input* model, misalnya jumlah transaksi 10 menit terakhir. Dihitung oleh *pipeline* kita.
- **Label**: jawaban benar untuk melatih, misalnya "transaksi ini terbukti *fraud*". Sering datang **terlambat** (hari sampai minggu, setelah investigasi).
- ***Training-serving skew***: *feature* saat melatih dihitung berbeda dengan *feature* saat model dipakai. Model bagus di uji, buruk di produksi.

Penyebab *skew* yang sering

- *Feature* latih ditulis di Spark SQL, *feature* produksi ditulis ulang di Java
- *Point-in-time* salah: *feature* latih memakai data yang belum ada saat transaksi terjadi (*data leakage*)
- Perbedaan zona waktu atau satuan

Ringkasan

Obatnya sama dengan masalah *hot–cold*: definisi *feature* sekali, *feature store online* dan *offline* (pertemuan 12).

Sumber: Kaitan dengan mata kuliah Machine Learning: evaluasi model dan *generalization*.

Eksperimen A/B

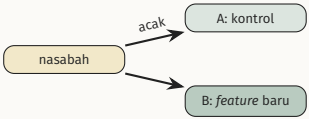
Mengapa korelasi tidak cukup

Tim produk melihat: nasabah yang memakai *feature* “tabungan tujuan” punya saldo rata-rata lebih tinggi. Apakah *feature* itu menaikkan saldo?

- Mungkin ya.
- Mungkin nasabah bersaldo tinggi memang lebih rajin mencoba *feature* baru (*selection bias*).
- Mungkin keduanya didorong oleh faktor lain, misalnya waktu gajian.

Eksperimen terkontrol

Bagi nasabah secara **acak** menjadi dua kelompok. Kelompok B mendapat *feature*, kelompok A tidak. Karena pembagiannya acak, perbedaan hasil yang cukup besar dapat diatribusikan pada *feature*.



Merancang eksperimen sebelum menjalankannya

1. **Hipotesis**: “Notifikasi pengingat menaikkan tingkat aktivasi 30 hari.”
2. ***Metric* utama** (satu) dan ***guardrail*** (misalnya rasio berhenti berlangganan notifikasi, keluhan).
3. **Unit randomisasi**: nasabah, bukan sesi. Pengacakan per sesi membuat satu nasabah melihat dua versi.
4. **Efek minimum yang bermakna** (MDE): kenaikan terkecil yang layak secara bisnis, misalnya 2 poin persentase.
5. **Ukuran sampel dan durasi** ditetapkan di awal, mencakup minimal satu siklus mingguan (dan siklus gajian bila relevan).
6. **Rencana analisis** ditulis sebelum data dilihat.

Ringkasan

Eksperimen yang tidak dirancang di awal hampir selalu berakhir dengan “mencari-cari” *metric* yang kebetulan signifikan.

Ukuran sampel: rumus kasar

Untuk membandingkan dua proporsi dengan $\alpha = 0{,}05$ (α dibaca “alfa”: peluang *false positive*, dua sisi) dan *power* 80%, aproksimasi yang sering dipakai:

$$n \approx \frac{16\, p(1-p)}{\delta^2} \quad \text{per kelompok}$$

dengan p = proporsi dasar dan δ (dibaca “delta”) = selisih minimum yang ingin terdeteksi.

Sumber: Kohavi, Tang, dan Xu, *Trustworthy Online Controlled Experiments*, 2020 (aturan praktis; angka 16 berasal dari $2(1{,}96 + 0{,}84)^2 \approx 15{,}7$).

Contoh hitungan

Aktivasi saat ini $p = 0{,}40$; ingin mendeteksi naik 2 poin, $\delta = 0{,}02$:

$$n \approx \frac{16 \times 0{,}4 \times 0{,}6}{0{,}02^2} = \frac{3{,}84}{0{,}0004} = 9.600$$

per kelompok, jadi sekitar 19.200 nasabah baru. Bila pembukaan rekening 1.000 per hari, eksperimen butuh sekitar 3 pekan.

Pelajaran

Separuhkan δ, sampel naik empat kali. Efek kecil butuh lalu lintas besar.

Membaca hasil: signifikansi dan selang kepercayaan

- ***p-value***: peluang melihat selisih sebesar ini (atau lebih) **bila sebenarnya tidak ada efek**. Bukan peluang bahwa hipotesis benar.
- **Selang kepercayaan 95%**: rentang nilai efek yang masuk akal. Lebih informatif daripada “signifikan atau tidak”.
- **Signifikan secara statistik** belum tentu **bermakna secara bisnis**: naik 0,1 poin bisa signifikan bila sampelnya jutaan.



Tiga selang kepercayaan ilustrasi.

Jebakan yang sering terjadi

Peeking

Melihat hasil setiap hari dan berhenti begitu p < 0,05. Dengan cukup sering mengintip, “signifikan” hampir pasti muncul walau tidak ada efek. Tetapkan durasi di awal, atau pakai metode *sequential testing* yang memang dirancang untuk itu.

Novelty effect

Nasabah mencoba *feature* karena baru, lalu kembali ke kebiasaan lama. Lihat tren efek per pekan, jangan hanya total.

Sample ratio mismatch

Direncanakan 50:50, yang tercatat 52:48. Biasanya tanda *bug* pada pengacakan atau pencatatan *event*. Periksa sebelum membaca hasil.

Terlalu banyak *metric*

Menguji 20 *metric* pada $\alpha = 0{,}05$ berarti sekitar satu *metric* “signifikan” karena kebetulan. Tetapkan satu *metric* utama.

Interferensi

Feature transfer antar nasabah: kelompok A ikut terpengaruh oleh B. Pertimbangkan pengacakan per kelompok atau wilayah.

Data product dan *data mesh*

Data product: data yang diperlakukan seperti produk

Sebuah *data product* punya pemilik, pengguna, dan janji. Contoh: “Transaksi harian terkonformasi” milik tim *core banking*.

- **Mudah ditemukan**: terdaftar di katalog
- **Dapat dipahami**: *schema*, definisi, contoh *query*
- **Dapat dipercaya**: SLO kesegaran dan *data quality*, uji otomatis
- **Dapat diakses dengan aman**: kebijakan akses jelas
- **Dapat dipakai bersama**: format terbuka, kunci yang konsisten

Data mesh (Dehghani): empat prinsip

1. *Domain ownership*: tim domain memiliki datanya
2. *Data as a product*
3. *Self-serve data platform*: tim platform menyediakan alat
4. *Federated computational governance*: aturan bersama yang ditegakkan otomatis

Sumber: Dehghani, *Data Mesh*, O’Reilly, 2022.

Kapan *data mesh* masuk akal

Cocok bila

- Banyak domain dengan tim *engineer* sendiri
- Tim data pusat menjadi hambatan (*queue* permintaan panjang)
- Platform swalayan sudah cukup matang

Belum cocok bila

- Organisasi kecil dengan satu tim data
- Tim domain belum punya kemampuan *data engineering*
- *Governance* dan katalog belum ada, *mesh* menjadi *mess*

Ringkasan

Gagasan *data product* berguna di organisasi sebesar apa pun. *Data mesh* penuh menuntut perubahan struktur organisasi dan kepemilikan data; memasang alat saja tidak cukup.

Literasi data dan bias

Data bisa menyesatkan orang yang jujur

- ***Survivorship bias***: menganalisis hanya nasabah yang bertahan, lalu menyimpulkan apa yang membuat nasabah bertahan.
- **Paradoks Simpson**: tren pada tiap cabang bisa berbalik arah ketika semua cabang digabung, karena komposisi kelompok berbeda.
- **Rata-rata menyembunyikan sebaran**: rata-rata saldo naik karena segelintir nasabah besar; median tidak berubah.
- **Bias pada model**: data historis pembiayaan bisa membawa pola penolakan masa lalu terhadap kelompok tertentu.

Kebiasaan *data engineer* yang membantu

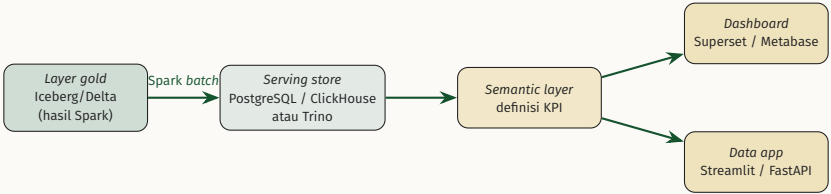
- Sajikan median dan persentil, bukan hanya rata-rata
- Sediakan dimensi untuk memecah angka (segmen, cabang, kanal)
- Dokumentasikan siapa yang **tidak** ada di data
- Tandai angka sementara vs final

Ringkasan

Amanah data juga berarti tidak membiarkan angka disalahpahami.

Membangun *dashboard*

Dari *layer gold* ke layar pengambil keputusan



- Alat BI tidak membaca *layer bronze* atau *silver*: hanya *gold* yang sudah bersih, PII-nya sudah di-*tokenize*, dan berbentuk *star schema*.
- Untuk agregat kecil, salin *gold* ke PostgreSQL; untuk miliaran baris dan *query* interaktif, pakai ClickHouse; untuk membaca Iceberg langsung, pakai Trino (pertemuan 12).
- *Dashboard* yang lambat tidak akan dipakai: targetkan *query* di bawah beberapa detik.

Superset atau Metabase?

	Apache Superset	**Metabase**
Kekuatan	visualisasi kaya, SQL Lab, banyak sumber data, terbuka penuh	sangat mudah, pengguna non-teknis bisa bertanya tanpa SQL
Pemasangan	beberapa layanan (*web*, *worker*, *metadata DB*, *cache*)	satu *container* sudah cukup untuk mencoba
Koneksi	SQLAlchemy: PostgreSQL, ClickHouse, Trino, DuckDB, dll.	driver bawaan: PostgreSQL, ClickHouse, Trino, dll.
Row-level access	ada *feature Row Level Security* di versi terbuka	*sandboxing* data umumnya *feature* edisi berbayar (cek edisi)
Cocok untuk	tim data, analis yang fasih SQL	manajemen dan unit bisnis

Untuk tugas besar, pilih salah satu. Yang dinilai adalah kebenaran angka dan kegunaan layar, bukan banyaknya *chart*.

Memasang Metabase dan menghubungkannya ke *gold*

```
services:
  postgres-gold:
    image: postgres:16
    environment:
      POSTGRES_PASSWORD: lab
      POSTGRES_DB: gold
    ports: ["5432:5432"]
  metabase:
    image: metabase/metabase:latest
    ports: ["3000:3000"]
    depends_on: [postgres-gold]
```

Superset: pakai *docker compose* resmi dari repositori Superset (port 8088). Versi *image* dikunci di *file* kelas.

1. Spark menulis tabel agregat *gold* ke PostgreSQL lewat JDBC (`df.write.jdbc(...)`) setiap selesai *batch*.
2. Buka `localhost:3000`, buat akun admin, tambahkan koneksi PostgreSQL `gold`.
3. Buat akun koneksi khusus *read-only*: alat BI tidak perlu hak tulis.
4. Tandai kolom tanggal dan nominal dengan tipe yang benar agar filter dan *aggregation* berjalan.

Anatomi *dashboard* yang dipakai

- **Satu *dashboard*, satu audiens, satu keputusan**: kepala cabang tidak butuh layar yang sama dengan tim *fraud*.
- **Atas: angka KPI** (*big number*) beserta pembanding (pekan lalu, target).
- **Tengah: tren** (*line chart*) untuk melihat arah.
- **Bawah: rincian** (tabel) untuk menelusuri.
- ***Filter*** global: tanggal, cabang, produk akad.
- ***Drill-down***: dari wilayah ke cabang ke transaksi, dengan hak akses yang sesuai.

Kesegaran dan *caching*

- Tampilkan "data per ..." di pojok layar, diambil dari `max(updated_at)`.
- *Cache query* sesuai SLO kesegaran: data harian cukup di-*cache* berjam-jam; data *hot* beberapa detik atau tanpa *cache*.
- *Auto-refresh* hanya untuk layar yang memang dipantau terus.

Row-level access

Kepala cabang Bogor hanya melihat Bogor. Terapkan di *serving store* (RLS PostgreSQL, pertemuan 14) atau di alat BI, jangan hanya menyembunyikan *filter*.

Visualisasi dan *storytelling*

Pilih *chart* dari pertanyaannya

Pertanyaan	***Chart***	**Contoh bank syariah**
Berapa angkanya sekarang?	*big number* + pembanding	pembiayaan cair hari ini
Bagaimana trennya?	*line chart*	nasabah aktif per pekan
Siapa paling besar atau kecil?	*bar chart* terurut	10 cabang dengan NPF tertinggi
Bagaimana sebarannya?	histogram, *box plot*	nominal transaksi per segmen
Bagian dari keseluruhan?	*stacked bar* (hindari *pie* > 4 irisan)	porsi akad murabahah, musyarakah, ijarah
Di mana?	peta	sebaran agen per kabupaten
Hubungan dua besaran?	*scatter plot*	umur rekening vs saldo rata-rata

Chart yang menyesatkan, dan *storytelling* yang jujur

Jebakan yang sering terlihat

- Sumbu Y *bar chart* tidak mulai dari nol: selisih kecil tampak raksasa.
- Dua sumbu Y dengan skala berbeda: korelasi palsu.
- Warna terlalu banyak; merah dan hijau saja (buta warna).
- Rata-rata tanpa sebaran; persentase tanpa jumlah sampel.
- Periode dipilih agar terlihat bagus (*cherry-picking*).

Storytelling tiga langkah

1. **Konteks**: apa pertanyaannya dan mengapa penting sekarang.
2. **Temuan**: satu pesan per layar, ditulis sebagai judul (“NPF cabang Depok naik dua bulan berturut-turut”), bukan “Grafik NPF”.
3. **Tindakan**: apa yang disarankan dan bagaimana mengukurnya.

Sumber: Prinsip umum visualisasi data; lihat juga Cole Nussbaumer Knaflic, *Storytelling with Data*.

Membangun *data app*

Kapan *dashboard* tidak cukup

Dashboard bagus untuk **melihat**. *Data app* dibutuhkan ketika pengguna perlu **berinteraksi**:

- memasukkan data: analis memberi label "*fraud*/bukan" untuk melatih model;
- simulasi: "berapa angsuran bila margin murabahah naik 1%?";
- alur kerja: menyetujui, menandai, mengunduh *file* laporan;
- memanggil model ML dan menampilkan alasannya.

Dua jalur

Streamlit: aplikasi *web* hanya dengan Python, cepat untuk prototipe dan alat internal.

FastAPI + *frontend*: API terpisah dari tampilan; lebih rapi untuk aplikasi yang dipakai banyak orang, butuh otentikasi, atau dipanggil sistem lain.

Streamlit: *filter*, *metric*, dan *chart* di atas *gold*

```python
import duckdb, streamlit as st

st.set_page_config(page_title="KPI Cabang", layout="wide")

@st.cache_resource           # satu koneksi untuk semua sesi
def db():
    return duckdb.connect("gold.duckdb", read_only=True)

@st.cache_data(ttl=300)      # hasil di-cache 5 menit
def harian(cabang):
    q = """SELECT tanggal, sum(nominal) AS nominal
           FROM fct_transaksi_harian WHERE cabang = ?
           GROUP BY tanggal ORDER BY tanggal"""
    return db().cursor().execute(q, [cabang]).df()

cabang = st.sidebar.selectbox("Cabang",
                              ["Bogor", "Bandung", "Depok"])
df = harian(cabang)
st.metric("Nominal 30 hari", f"Rp{df.nominal.tail(30).sum():,.0f}")
st.line_chart(df, x="tanggal", y="nominal")
```

- Jalankan: `streamlit run app.py` (port 8501).
- `gold.duckdb` adalah salinan agregat *gold* hasil Spark; bisa juga membaca Parquet atau Iceberg langsung.
- `cache_resource` untuk objek mahal (koneksi, model); `cache_data` untuk hasil *query*.
- Parameter *query* (`?`) mencegah SQL *injection*.

Streamlit: *form input* dan batasnya

```
with st.form("label_fraud"):
    trx_id = st.text_input("ID transaksi")
    label = st.radio("Label analis",
                     ["fraud", "bukan fraud"])
    alasan = st.text_area("Alasan singkat")
    kirim = st.form_submit_button("Simpan")

if kirim:
    simpan_label(trx_id, label, alasan,
                 analis=st.session_state["user"])
    st.success("Label tersimpan")
```

- Input disimpan ke tabel operasional (PostgreSQL), **bukan** menimpa *layer gold*; *pipeline* membacanya pada *batch* berikutnya.
- Catat siapa dan kapan (*audit trail*, pertemuan 14).
- Streamlit menjalankan ulang *script* setiap interaksi: pindahkan kerja berat ke *cache* atau ke Spark.
- Otentikasi: pasang di depan aplikasi (SSO / *reverse proxy*); jangan buka ke internet tanpa login.

Alternatif: FastAPI + *frontend* sederhana

```python
from fastapi import FastAPI, Depends
import duckdb

app = FastAPI()
con = duckdb.connect("gold.duckdb", read_only=True)

@app.get("/api/kpi/{cabang}")
def kpi(cabang: str, user=Depends(cek_token)):
    izinkan_cabang(user, cabang)   # row-level access
    q = """SELECT tanggal, sum(nominal) AS nominal
           FROM fct_transaksi_harian WHERE cabang = ?
           GROUP BY tanggal ORDER BY tanggal"""
    return con.cursor().execute(q, [cabang]) \
              .df().to_dict(orient="records")
```

- API mengembalikan JSON; *frontend* (HTML + JavaScript dengan pustaka *chart*, atau React) menggambarnya.
- Dokumentasi OpenAPI otomatis di `/docs` (kontrak, Sister P4).
- Cocok bila data dipakai juga oleh aplikasi lain atau butuh kendali akses rinci.
- Lebih banyak kode daripada Streamlit; pilih bila memang perlu.

Dashboard, *data app*, atau laporan?

	Dashboard	***Data app***	**Laporan (PDF/*notebook*)**
Tujuan	memantau KPI berkala	berinteraksi, input, simulasi	menjelaskan satu temuan
Pengguna	manajemen, unit bisnis	analis, petugas operasional	direksi, regulator
Kesegaran	menit–harian	sesuai kebutuhan	*snapshot* satu waktu
Usaha membangun	rendah (alat BI)	sedang–tinggi (kode)	rendah–sedang
Contoh bank	KPI cabang harian	pelabelan *fraud*, simulasi angsuran	laporan bulanan OJK

Aturan praktis: mulai dari *dashboard*; bangun *data app* hanya bila ada interaksi yang tidak bisa dilayani alat BI.

Persiapan demo UAS

Yang harus terlihat saat demo

- Satu *dashboard* atau *data app* yang menjawab pertanyaan bisnis yang ditulis di rancangan P12
- Angka di layar sama dengan hasil *query* langsung ke *layer gold*
- Label kesegaran data (“data per …”)
- Akses dibatasi sesuai peran (minimal dua peran)
- Jejak dari layar ke *pipeline*: tabel *gold* mana, *job* Spark mana

Skenario demo lima menit

1. Pertanyaan bisnis dan KPI-nya (30 detik).
2. Jalankan *job* Spark kecil; tunjukkan data baru masuk ke *gold*.
3. *Refresh* layar; angka berubah sesuai.
4. Masuk sebagai peran lain; data yang terlihat berbeda.
5. Satu keputusan yang bisa diambil dari layar ini.

When to use dan *best practice*

When to use: alat untuk *layer* keputusan

Alat	Pakai bila	Pertimbangkan alternatif bila
dbt *Semantic Layer*	model sudah di dbt; ingin *metric* sebagai kode di Git	tidak memakai dbt sama sekali
Cube	butuh API *metric* untuk aplikasi dan banyak alat BI, *caching*	cukup satu alat BI dengan model bawaannya
Metabase	*dashboard* swalayan cepat untuk pengguna non-teknis	butuh visualisasi dan izin yang sangat rinci
Superset	*dashboard* lebih kaya, banyak sumber data, terbuka	tim kecil yang ingin pemasangan paling sederhana
Streamlit	*data app* internal cepat, prototipe, input analis	aplikasi publik dengan banyak pengguna dan kendali akses rinci
FastAPI + *frontend*	data dipakai aplikasi lain, otentikasi dan izin rinci	hanya butuh layar pantau (alat BI cukup)
Python (statsmodels, scipy)	analisis eksperimen, uji hipotesis	volume eksperimen besar: platform eksperimen khusus
Platform eksperimen (GrowthBook, dll.)	banyak eksperimen paralel, *feature flag*, analisis otomatis	baru satu–dua eksperimen per kuartal

Best practice industri

1. **Mulai dari keputusan**, bukan dari data yang tersedia.
2. **Satu *metric*, satu definisi, satu pemilik**; definisi di Git.
3. **Setiap KPI punya *guardrail*.**
4. ***Metric* dihitung dari *layer gold***; tabel mentah tidak dipakai untuk *metric*.
5. ***Dashboard* punya tanggal kesegaran** yang terlihat jelas.
6. **Eksperimen dirancang di awal**: hipotesis, *metric*, sampel, durasi.
7. **Periksa *sample ratio mismatch*** sebelum membaca hasil.
8. **Laporkan selang kepercayaan**, bukan hanya *p-value*.
9. ***Feature* ML didefinisikan sekali** untuk latih dan produksi.
10. **Tinjau dampak keputusan otomatis** terhadap nasabah secara berkala.

Praktikum

Praktikum 1: KPI di *layer gold*

```sql
-- models/gold/kpi_aktivasi_harian.sql
with baru as (
  select nasabah_id, tgl_buka
  from {{ ref('dim_nasabah') }}
),
trx as (
  select nasabah_id, min(tgl_trx) as trx_pertama
  from {{ ref('fct_transaksi') }}
  where jenis = 'FINANSIAL'
    and dibuat_oleh = 'NASABAH'
  group by nasabah_id
)
select b.tgl_buka,
       count(*) as nasabah_baru,
       avg(case when t.trx_pertama
                  <= b.tgl_buka + interval 30 day
                then 1.0 else 0 end) as aktivasi_30h
from baru b left join trx t using (nasabah_id)
group by b.tgl_buka
```

1. Tambahkan model di atas ke proyek dbt dari pertemuan 6, atau tulis ulang sebagai *job* Spark atas *gold* pertemuan 9–10 (data sintetis).
2. Tambahkan uji: `not_null`, nilai `aktivasi_30h` antara 0 dan 1.
3. Definisikan tiga KPI: aktivasi 30 hari, NPF, waktu deteksi *fraud* (median dan P95).
4. Simpan hasilnya ke *serving store* (PostgreSQL) untuk praktikum 2.
5. Tuliskan definisi tiap KPI dalam satu kalimat yang dipahami orang bisnis.

Nasabah yang dibuka kurang dari 30 hari lalu belum boleh dihitung; tambahkan filter itu sendiri.

Praktikum 2: *dashboard* dan *data app*

Dashboard (Metabase atau Superset)

1. Hubungkan ke PostgreSQL `gold` dengan akun *read-only*.
2. Satu layar untuk kepala cabang: tiga KPI, satu tren, satu tabel rincian.
3. *Filter* tanggal dan cabang; label “data per …”.
4. Uji: angka KPI sama dengan *query* SQL langsung.

Data app (Streamlit)

1. Mulai dari contoh kode tadi; tambahkan pilihan rentang tanggal.
2. Tambahkan *form* pelabelan *fraud* yang menulis ke tabel terpisah.
3. Ukur waktu muat halaman dengan dan tanpa *cache*.
4. Tuliskan kapan Anda memilih *dashboard* dan kapan *data app*.

Latihan mandiri: menganalisis eksperimen A/B sintetis

```python
import numpy as np
from scipy.stats import chisquare
from statsmodels.stats.proportion import (
    proportions_ztest, confint_proportions_2indep)

rng = np.random.default_rng(42)
nA, nB = 9_700, 9_500            # data sintetis
aktifA = rng.binomial(1, 0.40, nA).sum()
aktifB = rng.binomial(1, 0.42, nB).sum()

# 1. sample ratio mismatch (rencana 50:50)
print(chisquare([nA, nB]).pvalue)

# 2. uji dua proporsi dan selang kepercayaan
z, p = proportions_ztest([aktifB, aktifA], [nB, nA])
lo, hi = confint_proportions_2indep(
    aktifB, nB, aktifA, nA, method="wald")
print(f"p={p:.3f}  CI selisih=[{lo:.3%}, {hi:.3%}]")
```

1. Jalankan kode; catat *p-value* SRM, *p-value* uji, dan selang kepercayaan.
2. Ulangi dengan `0.40` vs `0.40` (tidak ada efek) sebanyak 100 kali. Berapa kali “signifikan”?
3. Simulasikan *peeking*: uji setiap 1.000 nasabah tambahan dan berhenti saat p < 0,05. Berapa persen simulasi tanpa efek yang “berhasil”?
4. Tulis kesimpulan untuk direksi dalam tiga kalimat, termasuk ketidakpastiannya.

Tugas untuk pertemuan berikutnya

1. Kumpulkan laporan praktikum: definisi tiga KPI, tangkapan layar *dashboard*, tautan atau rekaman *data app*, dan (opsional) hasil latihan A/B.
2. Tugas besar UAS: tambahkan tabel KPI (nama, definisi, pemilik, *guardrail*, SLO kesegaran) dan sketsa layar *dashboard* atau *data app* ke rancangan.
3. Baca Dehghani, *Data Mesh*, Bab 1–4, dan Kohavi dkk., Bab 1–3.
4. Daftar semua kolom berisi data pribadi di *layer silver* dan *gold* proyek Anda (nama, NIK, nomor telepon, alamat, nomor rekening). Bawa daftarnya pekan depan.

Pertemuan depan

Data security: *encryption*, KMS, IAM, RBAC/ABAC, *row/column-level security*, *masking*, *tokenization*, dan audit log, mengamankan semua yang sudah kita bangun.

Terima kasih

Pertanyaan?