Pretest-Data-Analyst

Pretest by: Abdullah Mumtaz Fayyadh

DATA ANALYST – KNOWLEDGE BASE & USE CASE

A. KNOWLEDGE BASE

1. SQL Aggregation Understanding Sebuah tabel transaksi memiliki struktur sebagai berikut:

transaction_id | user_id | amount | transaction_date

Pertanyaan: Jelaskan perbedaan hasil antara:

SUM(amount)

SUM(DISTINCT amount)

Jawaban:

1. SUM(amount): Menjumlahkan semua nilai amount yang ada, mengakibatkan jika terdapat nilai yang sama muncul berulang, maka dihitung berulang juga nilainya.
2. SUM(DISTINCT amount): Menjumlahkan nilai amount secara unik (nilai yang sama dihitung sekali), contoh penggunaan: distinct user_id, untuk menghitung jumlah transaksi user. Perlu menggunakan distinct karena satu user bisa transaksi berkali-kali.

2. Data Architecture Understanding **Pertanyaan:**

- Apa perbedaan antara data source dan data warehouse?
- Mengapa dashboard atau laporan sebaiknya tidak langsung membaca data dari data source?

Jawaban:

1. Perbedaan antara data source dan data warehouse.
    - Data source adalah data operasional tempat data pertama kali tercatat/tersimpan.
    - Data warehouse adalah pusat penyimpanan data, yang merupakan gabungan data dari berbagai sumber dan sudah melalui proses pembersihan agar tersimpan secara konsisten.

2. Karena jika dashboard membaca langsung dari data source maka data belum "terstandarisasi" untuk analisis, rawan inkonsistensi, membebani sistem operasional (misal query berat SUM, GROUP BY, JOIN), isu kemanan, membuat perubahan schema yang tidak teratur sehingga menyebabkan dashboard error.

3. Error Analysis & Debugging **Pertanyaan:**

- Apa perbedaan antara error yang disebabkan oleh perubahan schema dan error karena query yang salah?
- Bagaimana cara membedakan kedua jenis error tersebut dalam praktik sehari-hari?

Jawaban:

1. Perbedaan antara error perubahan schema dan error karena query yang salah.
    - Schema error: Error yang terjadi karena perubahan struktur tabel, misalnya rename column, perubahan tipe data, tabel/dataset pindah.
    - Query error: Error yang terjadi karena query salah, misalnya typo syntax, group by tidak sesuai, salah fungsi agregasi.
2. Schema error biasanya pesannya seperti "column not found", "table nout found", "invalid schema". Sedangkan Query error pesannya seperti "syntax error near..", "invalid use of group function". Jika error terjadi padahal pernah menjalankan query yang sama, maka kemungkinan itu schema error. Yang terpenting adalah membaca pesan error dan lokasinya. Jika merasa ada perubahan struktur, cek perubahan schema menggunakan INFORMATION_SCHEMA, dan cek perubahan migrasi terakhir.

4. ETL Concept **Pertanyaan:**

- Jelaskan definisi dari ETL (Extract, Transform, Load) process.
- Sebutkan secara singkat fungsi dari masing-masing tahap ETL.

Jawaban:

1. ETL merupakan proses Extract (E) data dari berbagai sumber yang ingin dimasukkan ke data warehouse, Transfrom (T) mengolah data agar bersih dan sesuai kebutuh analisis, Load(L) memasukkan data yang bersih ke data warehouse.

2. Masing-masing tahap ETL
    - Extract (E): Mengambil data dari sumber: database aplikasi, API, file CSV. Yang dilakukan: memilih tabel/kolom yang dibutuhkan, ekstraksi, koneksi & autentikasi source.
    - Transfrom (T): Mengolah & merapikan data agar konsisten, dan sesuai kebutuhan, contoh transfrom:
        1. Cleaning: menghapus duplikat, handle null, validasi value
        2. Ubah tipe data
        3. Standarisasi: mata uang, timezone, penamaan kolom
        4. Join antar tabel, agregasi.
    - Load (L): Memasukkan data yang sudah siap ke data warehouse

5. BI Tool & Performance Awareness Kondisi: Sebuah dashboard Business Intelligence (BI) menjadi sangat lambat ketika dibuka oleh user. **Pertanyaan:**

- Sebutkan minimal tiga penyebab utama dashboard menjadi lambat.

Jawaban:

1. Query terlalu berat (join/agregasi masif, tanpa filter kalkulasi kompleks di BI)
2. Data warehouse tidak dioptimasi (tidak clustring data, baca raw tabel besar)
3. Dashboard terlalu banyak visual/filter yang memicu banyak query dan rendering.
4. Cache yang tidak efektif atau dimatikan, banyak user buka barengan menjadikan cache trashing.

B. USE CASE (STUDI KASUS)

Use Case 1: Data Belum Tersedia Kondisi: Anda diminta membuat visualisasi dashboard, namun data yang dibutuhkan belum tersedia atau belum dimigrasikan ke data warehouse.

Pertanyaan:

- Apa langkah pertama yang harus dilakukan?

Jawaban:

Langkah paling awal yang harus dilakukan adalah memastikan kebutuhan bisnis dan definisi data yang dibutuhkan (data requirements definition), misalnya memastikan dashboard ingin menunjukkan angka apa? total revenue atau jumlah transaksi? yang terpenting koordinasi dengan tim terkait (data engineer misalnya). Setelah semua definisi sesuai baru siapkan pipeline/migrasi ke data warehouse atau menyediakan extract sementara jika diperlukan.

Use Case 2: Perbedaan Nilai Database dan Dashboard Kondisi: Terdapat data transaksi di database dan di dashboard, namun nilai yang ditampilkan di dashboard berbeda dengan nilai di database.

Pertanyaan:

- Apa kemungkinan penyebab terjadinya perbedaan tersebut?
- Langkah apa yang harus dilakukan untuk memverifikasi dan menyelesaikan masalah ini?

Jawaban:

1. Kemungkinan penyebab: perbedaan definisi atau filter (status, refund), perbedaan periode & timezone, data dashboard belum refresh (ETL belum jalan/gagal), perbedaan sumber data, query salah yang menyebabkan double count.
2. Step by step:

   - Langkah 1: Menyamakan definisi & scope: tanggal, timezone, filter status

   - Langkah 2: Identifikasi tabel mana yang digunakan

   - Langkah 3: Run query yang sama dengan sumber yang sama

   - Langkah 4: Bandingkan output secara bertahap misal per hari atau status, hingga menemukan data transaksi yang mismatch

   - Langkah 5: Perbaiki akar masalah (refresh pipeline, koreksi query/join)

Use Case 3: Filter Tidak Sesuai Ekspektasi User Kondisi: User ingin memfilter data berdasarkan kondisi tertentu, namun hasil yang ditampilkan tidak sesuai atau data tidak muncul seperti yang diharapkan.

Pertanyaan:

- Apa langkah-langkah yang perlu dilakukan untuk menganalisis masalah ini?
- Bagaimana cara memastikan filter bekerja sesuai dengan logika data?

Jawaban:

1. Step by step analisis:

   - Langkah 1: Tanya ekspetasi user: User ingin memilih filter apa persisnya? misal range tanggal atau amount transaksi, user meminta "50 transaksi pada tanggal 21-28" apakah user berharap AND atau OR? Biasanya bukan kesalahan data, tapi kesalahan ekspetasi logika.

- Langkah 2: Sanity check sumber data: Memastikan nilai sesuai, misal user memilih "Jakarta" tetapi nilai "DKI Jakarta"

- Langkah 3: Periksa tipe data atau time zone, khususnya time zone

- Langkah 4: Cek logika filter: Misal AND atau OR, padahal kebanyakan data hanya memiliki salah satu kondisi

- Langkah 5: Periksa relation/join pada model BI

- Langkah 6: Jalankan tanpa filter, lalu mulai filter satu-satu

2. Cara memastikan filter bekerja sesuai dengan logika:

   - Tetapkan definisi filter yang jelas

   - Buat contoh case yang sesuai dengan tujuan

   - Lakukan validasi agregasi sederhana

   - Perbaiki baik di layer data (standarisasi) maupun di layer BI (relationship)

Use Case 4: Tipe Data Tidak Mendukung Aggregasi Kondisi: Tipe data pada database tidak sesuai untuk dilakukan agregasi di BI tools (misalnya data numerik tersimpan sebagai string).

Pertanyaan:

- Apa yang harus dilakukan untuk mengatasi masalah ini?
- Apa nama proses yang dilakukan untuk memperbaiki kondisi tersebut?

Jawaban:

1. Mengubah tipe data string menjadi numerik pada SQL/ETL, misalnya "10000" menjadi 10000 lalu validasi apakah sudah benar. Atau menambah kolom baru untuk numerik "amount_new" (hasil parsing), dan tetap membiarkan kolom string "amount"
2. Proses ini dinamakan Data cleaning atau Data tranformation pada proses ETL

Use Case 5: Nilai Terlalu Besar atau Terlalu Kecil Kondisi: Query tidak menghasilkan error, namun hasil perhitungan menunjukkan angka yang terlalu besar atau terlalu kecil.

Pertanyaan:

- Apa langkah-langkah validasi yang perlu dilakukan?
- Bagaimana cara memastikan bahwa hasil perhitungan sudah benar?

Jawaban:

1. Step by step validasi:

   - Langkah 1: Cek scope dan filter dasar: Apakah periode sesuai? timezone, status transaksi benar? filter WHERE kelupaan/terlalu luas

   - Langkah 2: Cek duplikasi: JOIN ke tabel yang lebih detail sehingga validasi lebih cepat, bandingkan count semua dengan count DISTINCT, cek GROUP BY

   - Langkah 3: Cek tipe data: Misal amount tersimpan cents tapi malah mengira rupiah

   - Langkah 4: Cek logika rumus: Misal refund harusnya dikurang tapi malah ditambah

   - Langkah 5: Spot check manual: Menggunakan sampel kecil, pilih 5-10 transaksi, hitung manual, memastikan output query sama dengan sampel

   - Langkah 6: Perbandingan dengan sumber lain: Bandingkan dengan dashboard lain, atau bandingkan total per hari, jika ada selisih maka fokus pada hari itu

2. Cara memastikan hasil perhitungan sudah sesuai yaitu dengan:

   1. Uji dari kecil ke besar

   2. Pastikan query benar pada sampel kecil

   3. Agregasi pada key yang tepat

   4. Tidak ada double count

   5. Memastikan BI sama (status, timezone)

   6. Terakhir, jalankan di full data dan sanity check.