Jawaban dari soal ChatGPT :

Soal Nomor 1

- Dari FD yang sudah ditemukan possibility candidate key dari FD tersebut adalah A
- Belum Mungkin masih berbentuk 2NF tetapi didalam itu masih ada Transitive Dependency yaitu di A -> C, C -> D, A -> D
- Buat table terpisah A(B, C), B(A,C,D), A bisa memanggil B dan C, serta A dan C bisa Memangil D

Soal Nomor 2

- Non key column hanya bergantung pada salah satu super key atau pun candidate key dan tidak full keynya
- Contoh : semisalkan {nim, kode_matkul} -> nama_mahasiswa, nama_matakuliah, sks, tetapi nim hanya bisa bergantung pada nim saja tidak pada kode_matkul nya

Soal Nomor 3

- Transitive dependency adalah bentuk dari anomaly Ketika terjadi indirect relation semisalkan X -> Y, Y -> Z, maka X -> Z, Contohnya itu nim -> kode_matakuliah, kode_matakuliah -> nama_matakuliah, maka nim -> nama_matakuliah,memecahnya dependency menjadi beberapa table agar tidak ada insert anomaly, update anomaly, dan delete anomaly

Soal Nomor 4

- StudentID -> student_name, major; CourseCode -> couserName, credit; studentID, CourseCode -> grade ini semua FDnya
- Super Key -> StudentId, CourseCode, ; Candidate Key -> StudentId, CourseCode
- 1 NF : StudentID -> student_name, major; CourseCode -> couserName, credit; studentID, CourseCode -> grade
- 2 NF : 1 NF Oke, Terdapat Partial Dependency -> StudentID -> student_name, major, dan CourseCode -> couserName, credit Solusi Pemecahan menjadi beberapa table : StudentID -> student_name, major, CourseCode -> couserName, credit, enroll -> studentId, CourseCode, grade
- 3NF : 2 NF Oke, No Transitive Dependency

Soal Nomor 5

- Reflexivity -> A -> ABC, A -> A, A -> B, A -> C
- Augmentation -> BC -> CD, B -> D
- Transitive -> A -> B, B -> C maka A -> C

Soal Nomor 6

➔ Table :

- Costumer (CostumerId(PK), CostumerName)
- Prodcut (ProductId(PK), product_name, quantity, price)
- Order(OrderId(PK), CostumerId(FK), ProductId(FK), order_date)

🧠 Pengantar Umum: Apa itu Database dan DBMS?

Database adalah kumpulan data yang terorganisir secara sistematis, yang dirancang agar dapat diakses, dikelola, dan diperbarui dengan mudah. Data dalam database biasanya berisi informasi yang saling berkaitan, seperti data mahasiswa, data penjualan, atau data pegawai. Untuk mengelola database, digunakan software khusus yang disebut **DBMS (Database Management System)**. DBMS bertanggung jawab mengatur penyimpanan data, pemrosesan query, pengamanan data, hingga pengendalian akses oleh banyak pengguna. Contoh DBMS populer antara lain MySQL, PostgreSQL, Oracle, dan MongoDB.

🏛️ Arsitektur DBMS

Arsitektur DBMS umumnya dibagi menjadi tiga lapisan utama: **internal level**, **conceptual level**, dan **external level**. Lapisan internal berkaitan dengan bagaimana data disimpan secara fisik di dalam storage. Conceptual level menggambarkan struktur logis dari database, misalnya tabel dan relasi antar entitas. Sementara itu, external level menyediakan tampilan spesifik bagi setiap pengguna atau aplikasi. Model ini disebut juga sebagai **three-schema architecture**, yang memberikan fleksibilitas dan keamanan karena pengguna tidak harus tahu bagaimana data disimpan secara fisik untuk bisa mengaksesnya.

🗃️ Jenis-Jenis Database

Database memiliki berbagai jenis berdasarkan model datanya. Yang paling umum adalah **Relational Database**, di mana data disimpan dalam bentuk tabel (relasi). Selain itu ada **Hierarchical Database** yang menyerupai struktur pohon, **Network Database** dengan struktur graph-like, serta **Object-Oriented Database** yang mengintegrasikan prinsip OOP. Di era modern, muncul juga **NoSQL Database** yang dirancang untuk skala besar dan fleksibilitas, terdiri dari key-value store, document store (seperti MongoDB), wide-column store (seperti Cassandra), dan graph database (seperti Neo4j).

🛠️ DDL dan DML (beserta Contohnya)

Di dalam DBMS, terdapat dua jenis perintah utama yang digunakan dalam SQL: **DDL (Data Definition Language)** dan **DML (Data Manipulation Language)**.

- **DDL** digunakan untuk **mendefinisikan struktur database**, seperti membuat tabel, menghapus tabel, atau mengubah skema. Contoh perintah DDL:

  ```
  CREATE TABLE Mahasiswa (
   NIM VARCHAR(10) PRIMARY KEY,
   Nama VARCHAR(100),
   Jurusan VARCHAR(50)
  );
  ```

  Penjelasan: Perintah di atas membuat sebuah tabel Mahasiswa dengan tiga kolom dan menetapkan NIM sebagai primary key.

- **DML** digunakan untuk **memanipulasi data** di dalam tabel, seperti menambahkan, mengubah, atau menghapus data. Contoh perintah DML:

  ```
  INSERT INTO Mahasiswa VALUES ('2201', 'Budi', 'Informatika');
  UPDATE Mahasiswa SET Nama = 'Budi Santoso' WHERE NIM = '2201';
  DELETE FROM Mahasiswa WHERE NIM = '2201';
  ```

  Penjelasan: Baris pertama menambahkan data mahasiswa, baris kedua memperbarui nama berdasarkan NIM, dan baris ketiga menghapus data mahasiswa dengan NIM 2201.

🔑 Kunci di Database (Database Keys)

Keys dalam database sangat penting karena digunakan untuk **mengidentifikasi setiap baris data secara unik**. Beberapa jenis key yang umum adalah:

- **Primary Key**: Kunci utama yang unik dan tidak boleh NULL.
- **Candidate Key**: Semua kunci potensial yang bisa jadi primary key.
- **Super Key**: Kombinasi atribut yang unik (termasuk candidate key + atribut tambahan).

- **Foreign Key**: Atribut yang merujuk ke primary key di tabel lain untuk menjaga hubungan antar tabel.

🧱 Schema Diagram

Schema diagram adalah representasi visual dari struktur database. Di dalamnya terdapat entitas (tabel), atribut (kolom), dan hubungan antar entitas. Diagram ini menggambarkan relasi antar tabel melalui garis penghubung—sering digunakan dalam ERD (Entity-Relationship Diagram). Contohnya:

- Entitas Mahasiswa punya atribut NIM, Nama
- Entitas MataKuliah punya atribut KodeMK, NamaMK
- Relasi Mengambil menghubungkan keduanya dengan atribut tambahan seperti Nilai

Setelah diterjemahkan ke bentuk relasional, kita mendapatkan tabel Mahasiswa, MataKuliah, dan KRS.

🧱 Schema Diagram

Schema diagram adalah representasi visual dari struktur database. Di dalamnya terdapat entitas (tabel), atribut (kolom), dan hubungan antar entitas. Diagram ini menggambarkan relasi antar tabel melalui garis penghubung—sering digunakan dalam ERD (Entity-Relationship Diagram). Contohnya:

- Entitas Mahasiswa punya atribut NIM, Nama
- Entitas MataKuliah punya atribut KodeMK, NamaMK
- Relasi Mengambil menghubungkan keduanya dengan atribut tambahan seperti Nilai

Setelah diterjemahkan ke bentuk relasional, kita mendapatkan tabel Mahasiswa, MataKuliah, dan KRS.

🔗 Relational Database dan Join Expression (dengan Contoh)

Relational Database menyimpan data dalam bentuk **relasi (tabel)**. Hubungan antar tabel dikelola menggunakan **foreign key** dan **JOIN** untuk penggabungan data.

JOIN Expression digunakan untuk **menggabungkan dua atau lebih tabel** berdasarkan kolom yang saling berelasi.

Contoh :

```
SELECT Mahasiswa.Nama, MataKuliah.NamaMK, Nilai.Nilai
FROM Mahasiswa
JOIN Nilai ON Mahasiswa.NIM = Nilai.NIM
JOIN MataKuliah ON Nilai.KodeMK = MataKuliah.KodeMK;
```

Penjelasan: Query di atas menggabungkan tiga tabel untuk menampilkan **nama mahasiswa, nama mata kuliah, dan nilai** yang diambil oleh mahasiswa tersebut. JOIN digunakan berdasarkan NIM dan KodeMK.

Jenis-jenis JOIN antara lain:

- **INNER JOIN**: hanya mengambil data yang cocok di kedua tabel.
- **LEFT JOIN**: semua data dari tabel kiri + yang cocok dari kanan.
- **RIGHT JOIN**: semua data dari tabel kanan + yang cocok dari kiri.
- **FULL JOIN**: semua data dari kedua tabel, cocok maupun tidak.

✅ Integrity Constraints

Integrity Constraints adalah aturan-aturan yang memastikan **validitas dan konsistensi data** dalam database. Contohnya:

- **NOT NULL**: kolom tidak boleh kosong.
- **UNIQUE**: nilai di kolom harus unik.

- **PRIMARY KEY**: unik dan tidak boleh NULL.
- **FOREIGN KEY**: menjaga hubungan antar tabel agar referensial tetap valid.
- **CHECK**: membatasi nilai yang boleh diinput, misalnya: