1. Apa yang dimaksud dengan LRS?

Logical Relational Schema (LRS) adalah representasi logis dari skema database dalam bentuk tabel relasional yang siap diimplementasikan dalam sistem manajemen basis data relasional (RDBMS). LRS merupakan hasil konversi dari **Entity-Relationship Diagram (ERD)** ke dalam bentuk tabel yang memiliki:

- **Atribut** sebagai kolom dalam tabel
- **Primary Key (PK)** sebagai identitas unik tiap baris data
- **Foreign Key (FK)** untuk menjaga hubungan antar tabel

2.Apa tujuan dari LRS? Mengapa ERD perlu dikonversikan ke LRS?

**Tujuan LRS:*

1. **Menerjemahkan ERD ke dalam bentuk tabel** yang dapat langsung diimplementasikan di database.
2. **Menentukan struktur relasional** untuk memastikan data terorganisir dengan baik.
3. **Memfasilitasi integritas data** dengan menerapkan primary key dan foreign key.
4. **Mengoptimalkan efisiensi penyimpanan** dengan menghindari duplikasi data.

Mengapa ERD harus dikonversikan ke LRS?

- ERD adalah model konseptual yang hanya menggambarkan hubungan antar entitas.
- LRS merupakan model logis yang dapat langsung digunakan dalam sistem basis data.
- LRS menyediakan skema yang lebih spesifik, termasuk definisi **tabel, atribut, primary key, dan foreign key**.

3. Jelaskan aturan dalam mengkonversikan ERD ke LRS! Berikan contoh pada setiap poin aturan!

Aturan Konversi:

1. **Setiap entitas menjadi tabel**
   - **Penjelasan:** Setiap entitas dalam ERD dikonversi menjadi tabel dalam LRS.
   - **Contoh:** Jika ada entitas *Mahasiswa* dengan atribut *NIM, Nama, Tanggal_Lahir*, maka tabel yang dibuat:

```
Mahasiswa(NIM (PK), Nama, Tanggal_Lahir)
```

2. **Atribut menjadi kolom dalam tabel**
   - **Penjelasan:** Atribut dalam ERD menjadi kolom dalam tabel pada LRS.
   - **Contoh:** Jika ada entitas *Dosen* dengan atribut *NIP, Nama, Jabatan*, maka tabelnya:

```
Dosen(NIP (PK), Nama, Jabatan)
```

3. **Hubungan (relationship) diterjemahkan sebagai Foreign Key**
   - **Penjelasan:** Jika ada hubungan antar entitas, maka foreign key ditambahkan dalam tabel yang sesuai.
   - **Contoh:** Jika entitas *Mahasiswa* memiliki hubungan dengan *Dosen* dalam bimbingan akademik:

```
Mahasiswa(NIM (PK), Nama, Tanggal_Lahir, NIP (FK))
```

4. **Hubungan Many-to-Many dikonversi menjadi tabel baru**
   - **Penjelasan:** Dalam hubungan many-to-many, dibuat tabel baru yang menghubungkan dua entitas.
   - **Contoh:** Jika *Mahasiswa* dan *Mata_Kuliah* memiliki hubungan many-to-many, maka dibuat tabel baru:

```
KRS(NIM (FK), Kode_MK (FK), Semester)
```

5. **Atribut multivalued dibuat sebagai tabel terpisah**
   - **Penjelasan:** Atribut yang memiliki lebih dari satu nilai per entitas dipisahkan ke dalam tabel baru.
   - **Contoh:** Jika entitas *Dosen* memiliki atribut multivalued *Nomor_Telepon*, maka dibuat tabel baru:

```
Nomor_Telepon_Dosen(NIP (FK), Nomor_Telepon)
```

---

4. Apa hubungan antara LRS dan Normalisasi? Berikan penjelasan disertai contohnya!

Hubungan antara LRS dan Normalisasi:

- Setelah ERD dikonversi ke LRS, skema tabel perlu dianalisis lebih lanjut menggunakan **normalisasi**.
- Normalisasi bertujuan untuk **mengurangi redundansi data** dan **mencegah anomali dalam manipulasi data**.

Contoh hubungan LRS dan Normalisasi:

Tanpa Normalisasi:

Sebelum normalisasi, tabel *Mahasiswa* mungkin memiliki bentuk berikut:

```
Mahasiswa(NIM, Nama, Alamat, Mata_Kuliah, Dosen)
```

Masalah:

- Jika seorang mahasiswa mengambil lebih dari satu mata kuliah, maka data **Nama dan Alamat** akan berulang.

Setelah Normalisasi (Minimal 3NF - Third Normal Form):

Tabel dipisah menjadi beberapa tabel untuk menghindari redundansi:

```
Mahasiswa(NIM (PK), Nama, Alamat)
Mata_Kuliah(Kode_MK (PK), Nama_MK)
KRS(NIM (FK), Kode_MK (FK))
```

Keuntungan setelah normalisasi:

- **Tidak ada duplikasi Nama dan Alamat dalam banyak baris**.
- **Struktur lebih fleksibel untuk perubahan data**.