

Tugas Besar Basis Data

KELOMPOK 11



Tim Kami



Halilah Roja Nasywa

121450046



Silvia Azahrani

121450070



Khairunnisa Rifda Aulia

121450091



Deyvan Loxefal

121450148



Deodry Siahaan

121450151

Daftar Isi



- | | | | |
|----------|--------------------------|----------|-----------------------------|
| 1 | Tentang Dataset | 5 | Data Manipulation Language |
| 2 | Normalisasi Tabel | 6 | Advance Query untuk Insight |
| 3 | Relational Schema | 7 | User Management |
| 4 | Data Definition Language | | |

Dataset Kami

Dataset diperoleh dari Kaggle. Berikut tautannya :

<https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database>

Dataset "Anime Recommendations Database" di Kaggle berisi informasi tentang anime dan rekomendasi pengguna. Dataset ini mencakup berbagai atribut anime, seperti judul, genre, skor, jumlah episode, durasi, studio produksi, dan banyak lagi.

Normalisasi

Normalisasi dilakukan untuk merancang struktur basis data agar menghilangkan anomali dan ambiguitas, serta meminimalkan redundansi data.

1. Normalisasi 1NF: Pastikan setiap atribut dalam setiap entitas memiliki nilai atomik (tidak ada nilai yang dapat dibagi-bagi). Jika ada atribut yang berisi multiple values, pecah atribut tersebut menjadi entitas baru.

	anime_id		name	type	episodes	rating
0	32281		Kimi no Na wa.	Movie	1	9.37
1	5114		Fullmetal Alchemist: Brotherhood	TV	64	9.26
2	28977		Gintama°	TV	51	9.25
3	9253		Steins;Gate	TV	24	9.17
4	9969		Gintama'	TV	51	9.16

2. Normalisasi 2NF: Pastikan bahwa setiap atribut non-kunci bergantung pada kunci primer secara keseluruhan dan tidak ada dependensi fungsional antara atribut non-kunci. Jika ada dependensi fungsional parsial, pisahkan atribut-atribut tersebut ke dalam entitas baru.

Tabel "User"

user_id	username	gender	age
1	John	Male	25
2	Emily	Female	30
3	Michael	Male	35
4	Sophia	Female	28
5	David	Male	22

Tabel "Rating"

rating_id	user_id	anime_id	rating
1	1	1	9.2
2	2	1	8.5
3	3	30276	7.8
4	4	457	6.5
5	5	32281	8.9

Tabel "Anime"

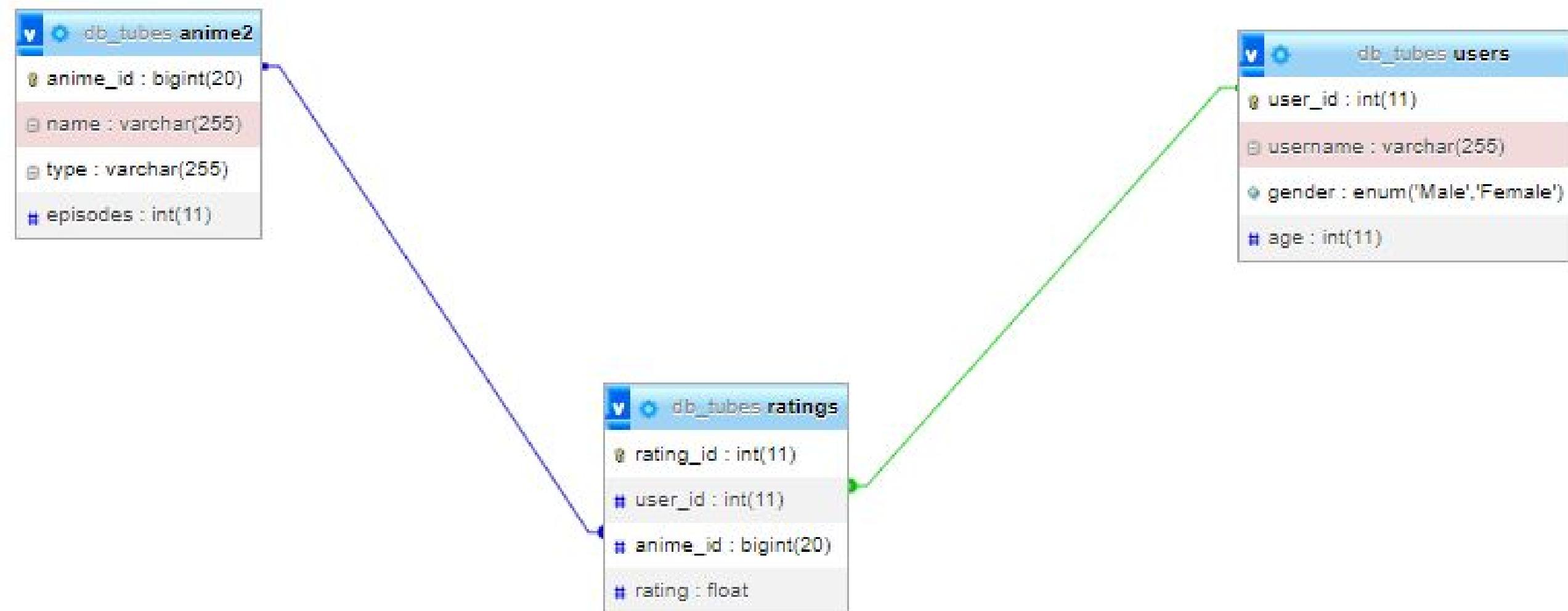
anime_id	name	type	episodes
1	Cowboy Bebop	TV	26
44	Rurouni Kenshin: Meiji Kenkaku Romantan - Tsuioku-hen	OVA	4
164	Mononoke Hime	Movie	1
199	Sen to Chihiro no Kamikakushi	Movie	1
263	Hajime no Ippo	TV	75

Relational Schema

Relational Schema adalah cara untuk merepresentasikan hubungan antara satu tabel dengan tabel lainnya melalui sebuah kolom kunci. Pada skema relasi sebuah primary key suatu tabel merupakan foreign key pada tabel lainnya.

Dalam Relational Schema juga terdapat beberapa jenis relasi yang mungkin ada antara tabel-tabel dalam database. Berikut adalah beberapa jenis relasi yang umum ditemukan dalam skema relasional:

- 1. One-to-One (Satu-ke-Satu):** Setiap baris dalam satu tabel hanya berhubungan dengan satu baris dalam tabel lainnya, dan sebaliknya.
- 2. One-to-Many (Satu-ke-Banyak):** Setiap baris dalam satu tabel berhubungan dengan banyak baris dalam tabel lainnya.
- 3. Many-to-Many (Banyak-ke-Banyak):** Setiap baris dalam satu tabel dapat berhubungan dengan banyak baris dalam tabel lainnya, dan sebaliknya.



Tabel anime2 dengan ratings berelasi one to many (1:n)
Tabel users dengan ratings berelasi one to many (1:n)

Data Definition Language

DDL merupakan komponen penting dalam SQL (Structured Query Language) yang digunakan untuk mengelola basis data. digunakan untuk mendefinisikan struktur dan skema dari basis data. DDL juga umumnya digunakan untuk membuat, mengubah, dan menghapus objek-objek database seperti tabel, indeks, view, dan constraint.

Pada tugas besar basis data kali ini kami menggunakan DDL diantaranya yaitu :

```
%sql # Normalisasi + Relation Schema dengan DDL

CREATE TABLE anime2(
    anime_id BIGINT(20) PRIMARY KEY,
    name VARCHAR(255),
    type VARCHAR(255),
    episodes INT(11)
);

CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(255),
    gender ENUM('Male', 'Female'),
    age INT
);

CREATE TABLE ratings (
    rating_id INT PRIMARY KEY,
    user_id INT,
    anime_id BIGINT(20),
    rating FLOAT,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (anime_id) REFERENCES anime2(anime_id)
);

* mysql+mysqldb://root@localhost
0 rows affected.
0 rows affected.
0 rows affected.

Out[20]:
[]
```

```
In [21]:
%sql SHOW TABLES;
* mysql+mysqldb://root@localhost
4 rows affected.

Out[21]:
Tables_in_db_tubes
anime
anime2
ratings
users
```

```
In [22]:
%sql DESC anime2;
* mysql+mysqldb://root@localhost
4 rows affected.

Out[22]:
Field      Type   Null  Key Default Extra
anime_id  bigint(20) NO   PRI  None
name       varchar(255) YES  None
type       varchar(255) YES  None
episodes   int(11)    YES  None
```

```
In [99]:
%sql DESC users;
* mysql+mysqldb://root@localhost
4 rows affected.

Out[99]:
Field      Type   Null  Key Default Extra
user_id   int(11)  NO   PRI  None
username  varchar(255) YES  None
gender    enum('Male','Female') YES  None
age       int(11)  YES  None
```

```
In [23]:
%sql DESC ratings;
* mysql+mysqldb://root@localhost
4 rows affected.

Out[23]:
Field      Type   Null  Key Default Extra
rating_id int(11) NO   PRI  None
user_id   int(11) YES  MUL  None
anime_id  bigint(20) YES  MUL  None
rating    float   YES  None
```

Data Manipulation Language

DML juga merupakan komponen penting dalam SQL (Structured Query Language) yang digunakan untuk memanipulasi data dalam basis data. DML digunakan untuk memasukkan, mengubah, menghapus, dan mengambil data dari tabel. Pada tugas besar basis data kali ini kami menggunakan DML diantaranya yaitu :

```
%sql # DML
INSERT INTO anime2 (anime_id, name, type, episodes)
SELECT anime_id, name, type, episodes
FROM anime;
* mysql+mysqldb://root@localhost
30 rows affected.

Out[25]:
[]
```

Kode di atas mengkopi data dari tabel "anime" ke tabel "anime2" menggunakan perintah INSERT INTO. Namun, perbedaannya adalah data yang dimasukkan berasal dari tabel "anime" dengan kolom-kolom yang sesuai: anime_id, name, type, dan episodes dengan pernyataan SELECT. Dengan menjalankan perintah ini, data dari tabel "anime" akan disalin ke tabel "anime2".

```
%sql # DML
SELECT * FROM anime2;
* mysql+mysqldb://root@localhost
30 rows affected.

Out[27]:
```

Kode di atas adalah perintah SQL untuk mengambil semua data dari tabel "anime2" menggunakan perintah SELECT * FROM anime2. Ini akan mengembalikan semua baris dan kolom data yang telah disalin dari tabel "anime" ke tabel "anime2" menggunakan pernyataan INSERT INTO sebelumnya.

```
%sql # DML
INSERT INTO users (user_id, username, gender, age) VALUES
(1, 'John', 'Male', 25),
(2, 'Emily', 'Female', 30),
(3, 'Michael', 'Male', 35),
(4, 'Sophia', 'Female', 28),
(5, 'David', 'Male', 22),
(6, 'Emma', 'Female', 26),
(7, 'Daniel', 'Male', 31),
(8, 'Olivia', 'Female', 29),
(9, 'Alexander', 'Male', 27),
(10, 'Ava', 'Female', 24),
(11, 'William', 'Male', 33),
(12, 'Mia', 'Female', 32),
(13, 'James', 'Male', 26),
(14, 'Isabella', 'Female', 27),
(15, 'Benjamin', 'Male', 28),
(16, 'Charlotte', 'Female', 25),
(17, 'Daniel', 'Male', 30),
(18, 'Amelia', 'Female', 31),
(19, 'Ethan', 'Male', 29),
(20, 'Sophia', 'Female', 26),
(21, 'Michael', 'Male', 27),
(22, 'Emily', 'Female', 28),
(23, 'Jacob', 'Male', 33),
(24, 'Olivia', 'Female', 32),
(25, 'Matthew', 'Male', 25),
(26, 'Ava', 'Female', 24),
(27, 'Joseph', 'Male', 29),
(28, 'Grace', 'Female', 28),
(29, 'Andrew', 'Male', 27),
(30, 'Emma', 'Female', 26);

* mysql+mysqldb://root@localhost
30 rows affected.

Out[28]:
```

Kode di atas adalah pernyataan SQL untuk memasukkan data ke dalam tabel "users". Perintah "INSERT INTO" digunakan untuk memasukkan baris-baris data ke dalam tabel. Dalam hal ini, kita memasukkan data ke dalam kolom-kolom "user_id", "username", "gender", dan "age" dalam tabel "users". Baris-baris data yang dimasukkan ditentukan dalam pernyataan "VALUES". Dalam kode di atas, kita memasukkan 30 baris data, dengan masing-masing baris terdiri dari nilai-nilai yang sesuai dengan kolom-kolom dalam urutan yang sama.

```
In [29]:
%sql #DML
SELECT * FROM users;
* mysql+mysqldb://root@localhost
30 rows affected.

Out[29]:
Kode di atas adalah perintah SQL untuk mengambil semua data dari tabel "users" menggunakan perintah SELECT * FROM users. Ini akan mengembalikan semua baris dan kolom data pengguna yang telah dimasukkan sebelumnya menggunakan pernyataan INSERT INTO.
```

```
In [103]: %sql # DML
INSERT INTO ratings (rating_id, user_id, anime_id, rating) VALUES
(1, 1, 1, 9.2),
(2, 2, 1, 8.5),
(3, 3, 30276, 7.8),
(4, 4, 457, 6.5),
(5, 5, 32281, 8.9),
(6, 6, 199, 7.2),
(7, 7, 1, 6.8),
(8, 8, 32281, 9.6),
(9, 9, 5114, 8.1),
(10, 10, 1, 7.4),
(11, 1, 457, 6.9),
(12, 2, 11061, 8.2),
(13, 3, 32281, 9.3),
(14, 4, 21939, 7.7),
(15, 5, 918, 8.4),
(16, 6, 32935, 7.6),
(17, 7, 32281, 9.0),
(18, 8, 44, 8.7),
(19, 9, 1, 6.3),
(20, 10, 1575, 7.9),
(21, 1, 12355, 9.1),
(22, 2, 199, 8.3),
(23, 3, 44, 7.5),
(24, 4, 1, 6.7),
(25, 5, 28977, 8.8),
(26, 6, 44, 7.4),
(27, 7, 457, 9.2),
(28, 8, 17074, 8.6),
(29, 9, 1, 7.0),
(30, 10, 1575, 7.3);

* mysql+mysqldb://root@localhost
30 rows affected.

Out[103]: []
```

Kode di atas adalah perintah SQL untuk memasukkan data ke dalam tabel "ratings" menggunakan perintah INSERT INTO. Data peringkat anime dimasukkan ke dalam tabel dengan nilai-nilai yang sesuai untuk kolom-kolom rating_id, user_id, anime_id, dan rating. Terdapat total 30 baris data yang dimasukkan ke dalam tabel.

```
In [31]:
%sql #DML
SELECT * FROM ratings;
* mysql+mysqldb://root@localhost
30 rows affected.

Out[31]:
```

Kode di atas adalah perintah SQL untuk mengambil semua data dari tabel "ratings" menggunakan perintah SELECT * FROM ratings. Ini akan mengembalikan semua baris dan kolom data peringkat anime yang telah dimasukkan sebelumnya menggunakan pernyataan INSERT INTO.

Insight Data

Dalam konteks basis data, **insight** mengacu pada pemahaman atau pengetahuan baru yang diperoleh dari analisis data. **Insight** merupakan informasi berharga yang dapat digunakan untuk mengambil keputusan atau memahami pola, tren, atau hubungan yang mungkin tidak terlihat secara langsung

Advanced query, atau kueri lanjutan, mengacu pada penggunaan fitur dan teknik yang lebih kompleks dalam mengeksekusi pertanyaan atau manipulasi data dalam basis data. Dibandingkan dengan kueri dasar, kueri lanjutan melibatkan pemahaman yang lebih mendalam tentang struktur dan fitur-fitur spesifik yang ada dalam sistem manajemen basis data (DBMS).

In [32]:

```
%sql # Advance Query  
  
# Menampilkan anime dengan rating tertinggi  
SELECT name, rating  
FROM anime  
ORDER BY rating DESC  
LIMIT 10;
```

* mysql+mysqldb://root@localhost

10 rows affected.

Out[32]:

	name	rating
	Kimi no Na wa.	9.37
	Fullmetal Alchemist: Brotherhood	9.26
	Gintama°	9.25
	Steins;Gate	9.17
	Gintama'	9.16
	Haikyuu!!: Karasuno Koukou VS Shiratorizawa Gakuen Koukou	9.15
	Hunter x Hunter (2011)	9.13
	Ginga Eiyuu Densetsu	9.11
	Gintama'; Enchousen	9.11
	Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare	9.1

In [33]:

```
%sql # Advance Query
```

```
# Menampilkan rata-rata rating untuk setiap type  
SELECT type, AVG(rating) AS average_rating  
FROM anime  
GROUP BY type  
LIMIT 10;
```

* mysql+mysqldb://root@localhost
3 rows affected.

Out[33]:

type	average_rating
Movie	8.987142857142858
OVA	8.969999999999999
TV	8.976190476190476

Code tersebut digunakan untuk mengambil rata-rata rating untuk setiap tipe anime dari tabel "anime". SELECT yang digunakan untuk memilih kolom "type" dan rata-rata rating ("AVG(rating)") dari tabel "anime". FROM yang menentukan tabel yang akan digunakan dalam query, yaitu tabel "anime". GROUP BY yang digunakan untuk mengelompokkan hasil berdasarkan kolom "type". LIMIT yang membatasi jumlah baris hasil yang akan ditampilkan

Code tersebut digunakan untuk mengambil 10 anime dengan rating tertinggi dari tabel "anime". SELECT yang digunakan untuk memilih kolom "name" dan "rating" dari tabel "anime". FROM yang menentukan tabel yang akan digunakan dalam query, yaitu tabel "anime". ORDER BY yang mengurutkan hasil berdasarkan kolom "rating" secara menurun (DESC). LIMIT yang membatasi jumlah baris hasil yang akan ditampilkan (10).

Insight

```
In [34]:  
%%sql # Advance Query  
  
# Menampilkan pengguna yang memberikan rating terendah  
SELECT u.username, r.rating  
FROM users u  
JOIN ratings r ON u.user_id = r.user_id  
ORDER BY r.rating ASC  
LIMIT 10;  
  
* mysql+mysqldb://root@localhost  
10 rows affected.  
  
Out[34]:  
  
username rating  
-----  
Alexander 6.3  
Sophia 6.5  
Sophia 6.7  
Daniel 6.8  
John 6.9  
Alexander 7.0  
Emma 7.2  
Ava 7.3  
Ava 7.4  
Emma 7.4
```

Code tersebut digunakan untuk mengambil 10 pengguna yang memberikan rating terendah dari tabel "users" dan "ratings". SELECT yang digunakan untuk memilih kolom "username" dari tabel "users" dan kolom "rating" dari tabel "ratings". FROM yang menentukan tabel "users" dan memberikan alias "u" untuk tabel tersebut. JOIN yang digunakan untuk menggabungkan tabel "users" dan "ratings" berdasarkan kolom "user_id" yang ada di kedua tabel. ORDER BY yang mengurutkan hasil berdasarkan kolom "rating" secara menaik (ASC) atau rating terendah ke tertinggi. LIMIT yang membatasi jumlah baris hasil yang akan ditampilkan

```
In [35]:  
%%sql # Advance Query  
  
# Menampilkan daftar anime yang belum mendapatkan rating  
SELECT a.name  
FROM anime2 a  
LEFT JOIN ratings r ON a.anime_id = r.anime_id  
WHERE r.rating_id IS NULL  
  
* mysql+mysqldb://root@localhost  
15 rows affected.  
  
Out[35]:  
  
name  
-----  
Mononoke Hime  
Hajime no Ippo  
Ginga Eiyuu Densetsu  
Tengen Toppa Gurren Lagann  
Code Geass: Hangyaku no Lelouch R2  
Clannad: After Story  
Suzumiya Haruhi no Shoushitsu  
Steins;Gate  
Gintama&#039;  
Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare  
Gintama&#039;; Enchousen  
Shigatsu wa Kimi no Uso  
Mushishi Zoku Shou 2nd Season  
Koe no Katachi  
Haikyuu!! Second Season
```

Code tersebut digunakan untuk mengambil daftar anime yang belum mendapatkan rating dari tabel "anime2" dan "ratings". SELECT yang digunakan untuk memilih kolom "name" dari tabel "anime2". FROM yang menentukan tabel "anime2" dan memberikan alias "a" untuk tabel tersebut. LEFT JOIN yang digunakan untuk menggabungkan tabel "anime2" dan "ratings" berdasarkan kolom "anime_id" yang ada di kedua tabel. WHERE yang digunakan untuk memfilter baris-baris di mana kolom "rating_id" dari tabel "ratings" memiliki nilai NULL

```
# Menampilkan anime dengan episode terbanyak  
SELECT name, episodes  
FROM anime  
ORDER BY episodes DESC  
LIMIT 10;  
  
* mysql+mysqldb://root@localhost  
10 rows affected.  
  
Out[36]:  
  
name episodes  
-----  
Gintama 201  
Hunter x Hunter (2011) 148  
Ginga Eiyuu Densetsu 110  
Hajime no Ippo 75  
Fullmetal Alchemist: Brotherhood 64  
Gintama° 51  
Gintama&#039; 51  
Tengen Toppa Gurren Lagann 27  
Cowboy Bebop 26  
Monogatari Series: Second Season 26
```

Code tersebut digunakan untuk mengambil 10 anime dengan jumlah episode terbanyak dari tabel "anime". SELECT yang digunakan untuk memilih kolom "name" dan "episodes" dari tabel "anime". FROM yang menentukan tabel yang akan digunakan dalam query, yaitu tabel "anime". ORDER BY yang mengurutkan hasil berdasarkan kolom "episodes" secara menurun (DESC) atau jumlah episode terbanyak ke terendah. LIMIT yang membatasi jumlah baris hasil yang akan ditampilkan

User Management

User Management adalah proses mengelola pengguna dan hak akses dalam sistem database. Tujuannya adalah untuk mengatur dan mengontrol akses pengguna ke database dan melindungi data dari pengguna yang tidak berwenang.

```
CREATE USER 'user_root'@'localhost';
CREATE USER 'user_1'@'localhost';
CREATE USER 'user_2'@'localhost';
```

Perintah ini untuk membuat pengguna baru dengan nama user_root, user_1, dan user_2 yang memiliki akses dari host 'localhost'.

Namun, perintah ini tidak memberikan hak akses atau izin apa pun kepada pengguna ini.

```
GRANT ALL PRIVILEGES ON db_tubes.* TO
'user_root'@'localhost';
```

Perintah ini memberikan semua hak akses (ALL PRIVILEGES) kepada 'user_root'@'localhost' pada database 'db_tubes'. Pengguna ini memiliki izin penuh untuk operasi seperti SELECT, INSERT, UPDATE, DELETE, DROP, dan ALTER pada tabel dan objek dalam database tersebut.

```
GRANT SELECT, INSERT, DELETE, DROP, ALTER ON db_tubes.* TO
'user_1'@'localhost';
```

Perintah ini memberikan hak akses SELECT, INSERT, DELETE, DROP, dan ALTER kepada 'user_1'@'localhost' pada database 'db_tubes'. Pengguna ini dapat membaca, menyisipkan, menghapus, menghapus tabel atau objek, dan mengubah struktur tabel atau objek dalam database tersebut.

```
GRANT SELECT, UPDATE, DELETE, DROP ON db_tubes.* TO
'user_2'@'localhost';
```

Perintah ini memberikan hak akses SELECT, UPDATE, DELETE, dan DROP kepada 'user_2'@'localhost' pada database 'db_tubes'. Pengguna ini dapat membaca, memperbarui, menghapus, dan menghapus tabel atau objek dalam database tersebut.



TERIMAKASIH

