

# Veri Tabanı

*Create Database, Primary Key*

Hüseyin Ahmetoğlu

# CREATE DATABASE

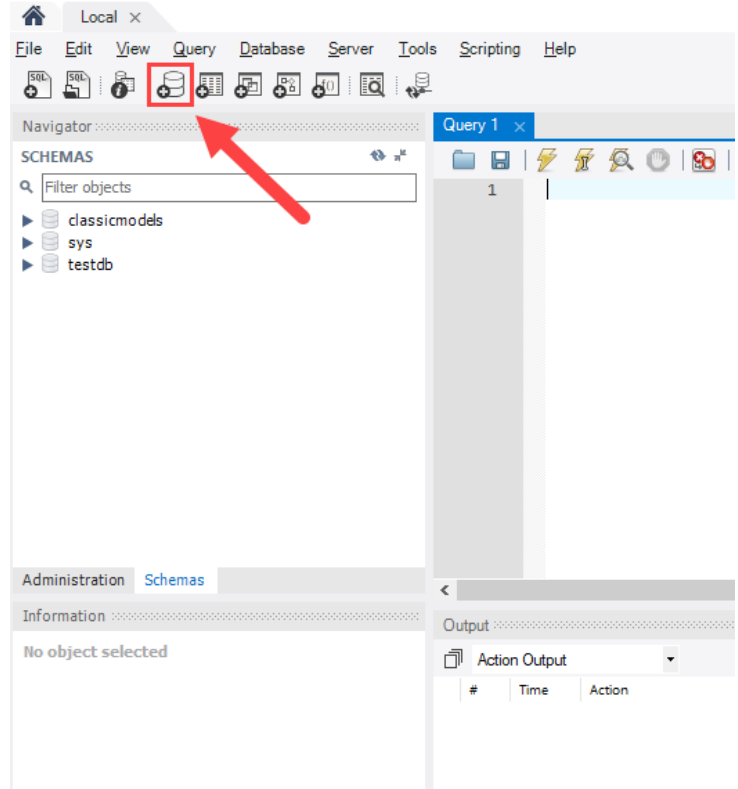
- ▶ MySQL, veri tabanındaki tablolara karşılık gelen tüm dosyaları içeren bir dizin olarak bir veri tabanı oluşturur.
- ▶ MySQL'de yeni bir veritabanı oluşturmak için,

```
CREATE DATABASE [IF NOT EXISTS] database_name  
[CHARACTER SET charset_name]  
[COLLATE collation_name]
```

- ▶ MySQL sunucu örneği içinde benzersiz olmalıdır. Zaten var olan bir ada sahip bir veritabanı oluşturmaya çalışırsanız, MySQL bir hata verir.


# MySQL Workbench kullanarak yeni bir veritabanı oluřturmak

- MySQL'de řema, veritabanı ile eřanlımlıdır. Yeni bir řema oluřturmak aynı zamanda yeni bir veritabanı oluřturmak anlamına gelir.



# MySQL Workbench kullanarak yeni bir veritabanı oluşturmak

Query 1 testdb2 - Schema x

 Name: testdb2 1 the name of the schema here. You can use

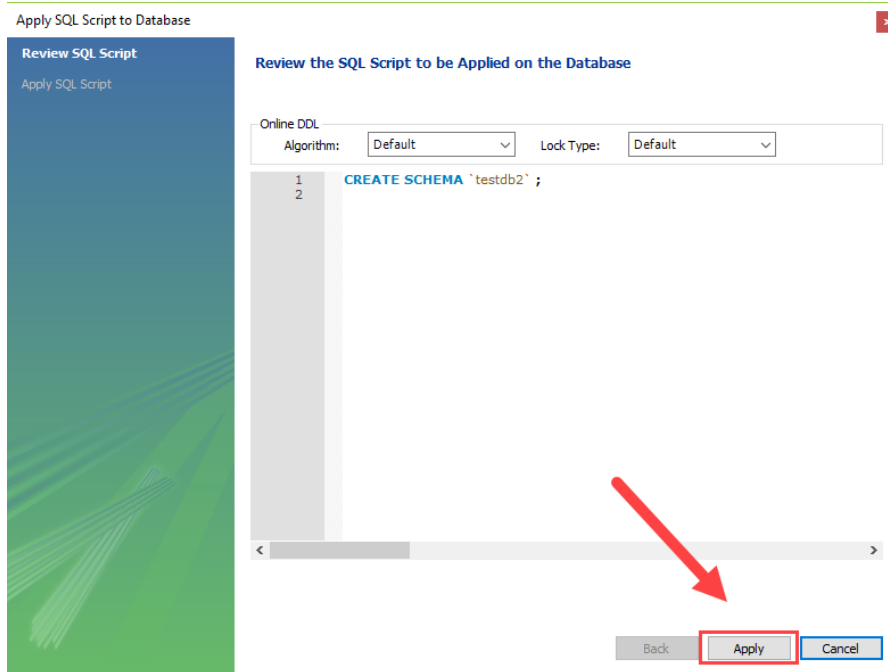
Rename References Refactor model, changing all references found in view, triggers, stored procedures and functions from the old

Charset/Collation: Default Charset Default Collation 2 character set and its collation selected here will

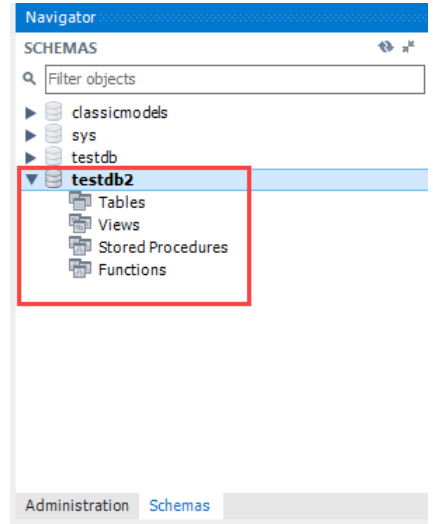
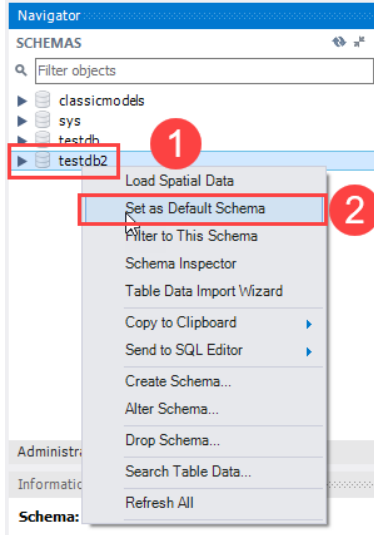
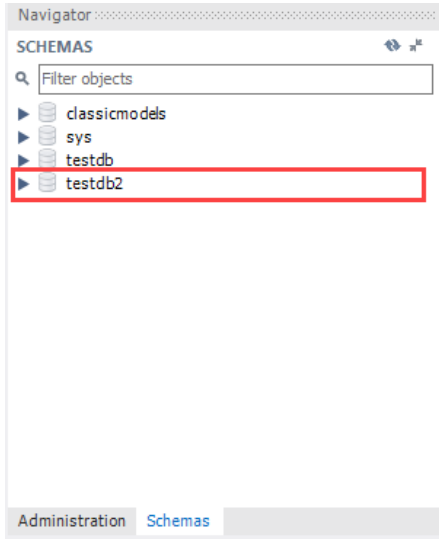
Schema

3 Apply Revert

# MySQL Workbench kullanarak yeni bir veritabanı oluşturmak



# MySQL Workbench kullanarak yeni bir veritabanı oluşturmak



# DROP DATABASE

- DROP DATABASE deyimi veri tabanındaki tüm tabloları ve veri tabanını kalıcı olarak siler. Bu nedenle, bu ifadeyi kullanırken çok dikkatli olmalısınız.

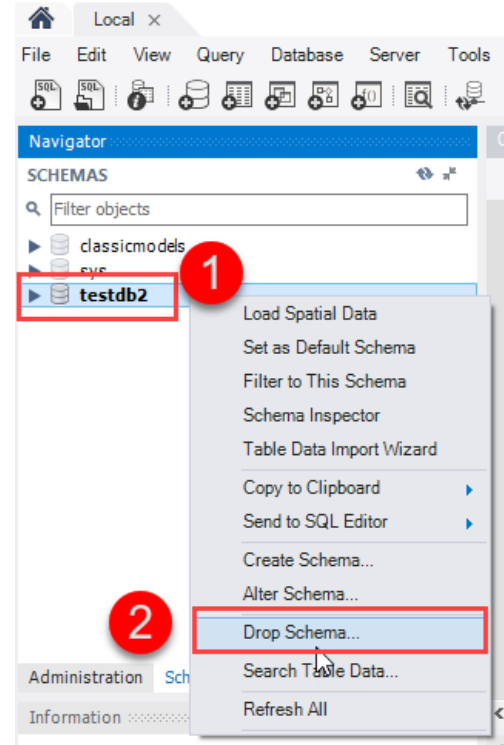
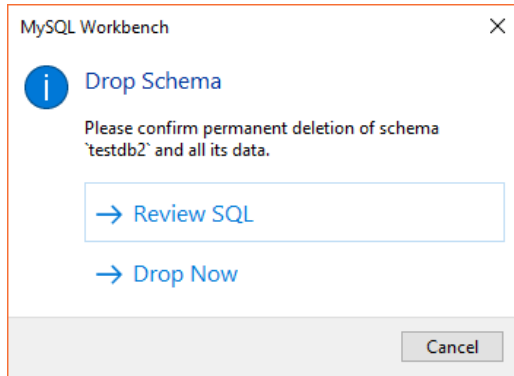
```
DROP DATABASE [IF EXISTS] database_name;
```

- Var olmayan bir veri tabanını sildiğinizde bir hatanın oluşmasını önlemek için, IF EXISTS seçeneği vardır. Bu durumda, MySQL herhangi bir hata vermeden ifadeyi sonlandırır.
- Aşağıdaki kod satırı da aynı amaç için kullanılabilir.

```
DROP SCHEMA [IF EXISTS] database_name;
```

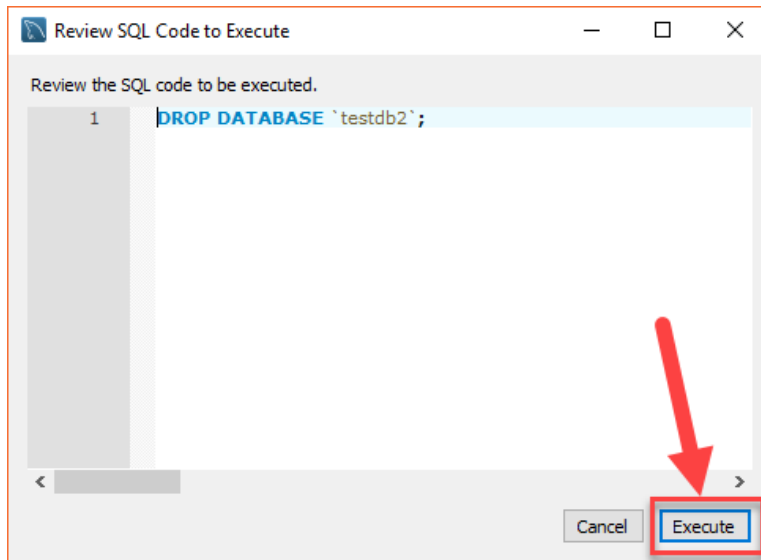
# DROP DATABASE

- MySQL Workbench, silme işlemini onaylamak için bir iletişim kutusu görüntüler. Eğer Review SQL seçilirse çalıştırılacak SQL ifadesi görülebilir. Drop Now ile veritabanı hemen kaldırılacaktır.





# DROP DATABASE



Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
✓ 1	18:19:38	DROP DATABASE `testdb2`	0 row(s) affected	0.015 sec

# MySQL Veri Tipleri Özet

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 2147483647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LOBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

# MySQL Depolama Motorları

- ▶ Depolama motorları, MYSQL'DE farklı tablo türleri için SQL işlemlerini yürüten MYSQL bileşenleridir. InnoDB, varsayılan ve en genel amaçlı depolama motorudur.
- ▶ Sunucunuuzn hangi depolama motorlarını desteklediğini belirlemek için SHOW ENGINES komutunu kullanabiliriz.

```
mysql> SHOW ENGINES;
```

Engine	Support	Comment	Transactions	XA	Savepoints
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MyISAM	DEFAULT	MyISAM storage engine	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
InnoDB	YES	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

9 rows in set (0.00 sec)

# MySQL Depolama Motorları

- Engine : depolama motorunun adını içerir.
- Support : sürücümüz üzerinde desteklenip desteklenmediği içerir.
- Comment : depolama motoru hakkında kısa bir açıklama içerir.
- Transactions : transaction yapısını destekleyip, desteklemediği bilgisini içerir.
- XA : Koordinat tabanlı transaction işlemlerini destekleyip, desteklemediği bilgisini içerir.
- Savepoints : Yapılan işlemlerin geri alınmasına (rollback) yönelik kayıt noktaları (savepoints) oluşturmayı destekleyip, desteklemediği bilgisini içerir.
- ▶ Depolama **motorları** belirli bir veritabanı **tablosundan bilgi çekmeyi tabloda bilgi saklamayı ve tablodaki bilgiyi kontrol etmeyi** sağlar.
- ▶ InnoDB, MySQL 5.5 sürümünden bu yana varsayılan depolama motoru haline geldi. InnoDB depolama motoru, ACID işlemi, bilgi bütünlüğü ve çökme kurtarma gibi ilişkisel bir veritabanı yönetim sisteminin birçok avantajını sağlar. Önceki sürümlerde MySQL, MyISAM'ı varsayılan depolama motoru olarak kullanıyordu.

# CREATE TABLE

- CREATE TABLE deyimi, bir veritabanında yeni bir tablo oluşturmanıza olanak sağlar.

```
CREATE TABLE [IF NOT EXISTS] table_name(  
    column_1_definition,  
    column_2_definition,  
    ...,  
    table_constraints  
) ENGINE=storage_engine;
```

# CREATE TABLE

- ▶ `column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT] column_constraint;`
- ▶ `column_name` sütunun adını belirtir. Her sütunun belirli bir veri türü ve isteğe bağlı boyutu vardır, ör. `VARCHAR(255)`
- ▶ `NOT NULL` kısıtlama ifadesidir ve sütunun `NULL` değer içermemesini sağlar. `NOT NULL` kısıtlaması yanında ek kısıtlamalara sahip olabilir `CHECK` ve `UNIQUE` .
- ▶ `DEFAULT` sütun için varsayılan bir değer belirtir.
- ▶ `AUTO_INCREMENT` yeni değerler tabloya eklendiğinde sütunun değerinin otomatik olarak bir artırıldığını gösterir. Her tabloda maksimum bir `AUTO_INCREMENT` bölümü olabilir.

# CREATE TABLE

```
CREATE TABLE IF NOT EXISTS tasks (  
    task_id INT AUTO_INCREMENT PRIMARY KEY,  
    title VARCHAR(255) NOT NULL,  
    start_date DATE,  
    due_date DATE,  
    status TINYINT NOT NULL,  
    priority TINYINT NOT NULL,  
    description TEXT,  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
) ENGINE=INNODB;
```

# CREATE TABLE

```
DESCRIBE tasks;
```

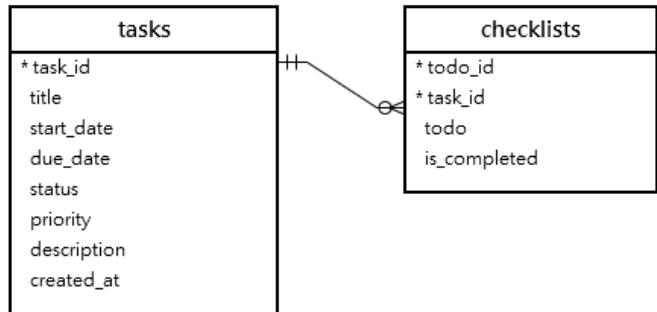
	Field	Type	Null	Key	Default	Extra
►	task_id	int(11)	NO	PRI	<small>NULL</small>	auto_increment
	title	varchar(255)	NO		<small>NULL</small>	
	start_date	date	YES		<small>NULL</small>	
	due_date	date	YES		<small>NULL</small>	
	status	tinyint(4)	NO		<small>NULL</small>	
	priority	tinyint(4)	NO		<small>NULL</small>	
	description	text	YES		<small>NULL</small>	
	created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT GENERATED

tasks
* task_id
title
start_date
due_date
status
priority
description
created_at



# CREATE TABLE

```
CREATE TABLE IF NOT EXISTS checklists (  
  todo_id INT AUTO_INCREMENT,  
  task_id INT,  
  todo VARCHAR(255) NOT NULL,  
  is_completed BOOLEAN NOT NULL DEFAULT FALSE,  
  PRIMARY KEY (todo_id , task_id),  
  FOREIGN KEY (task_id)  
    REFERENCES tasks (task_id)  
    ON UPDATE RESTRICT ON DELETE CASCADE  
);
```



# Primary Key

- ▶ Birincil anahtar, tablodaki her satırı benzersiz şekilde tanımlayan bir sütun veya sütun kümesidir. Birincil anahtar için şu kurallar geöçerlidir:
  - Birincil anahtar benzersiz değ erler içermelidir. Birincil anahtar birden çok s utundan olu uyorsa, bu s utunlardaki değ erlerin kombinasyonu benzersiz olmalıdır.
  - Birincil anahtar s utununun NULL olması durumunda g ncelleme ve silme gibi i lemler hatayla sonu lanacaktır. MySQL  rt k olarak PRIMARY KEY s utununu not null kısıtlaması getirir.
  - Bir tablonun yalnızca bir birincil anahtar olabilir.
- ▶ MySQL tamsayılarla daha hızlı  alı tı ından birincil anahtar s utununun veri tipi tam sayı olmalıdır,  r. INT, BIGINT. Ayrıca, birincil anahtar i in tamsayı t r n n değ er aralıklarının, tablonun sahip olabilece i t m olası satırları depolamak i in yeterli oldu undan emin olmalısınız.

# Primary Key

```
CREATE TABLE table_name(  
    primary_key_column datatype PRIMARY KEY,  
    ...  
);
```

```
CREATE TABLE table_name(  
    primary_key_column1 datatype,  
    primary_key_column2 datatype,  
    ...,  
    PRIMARY KEY(column_list)  
);
```

```
CREATE TABLE table_name (  
    primary_key_column datatype,  
    ... ,  
    PRIMARY KEY(primary_key_column)  
);
```

# Primary Key

```
CREATE TABLE users(  
    user_id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(40),  
    password VARCHAR(255),  
    email VARCHAR(255)  
);
```

```
CREATE TABLE roles(  
    role_id INT AUTO_INCREMENT,  
    role_name VARCHAR(50),  
    PRIMARY KEY(role_id)  
);
```

# Primary Key

```
CREATE TABLE user_roles(  
    user_id INT,  
    role_id INT,  
    PRIMARY KEY(user_id,role_id),  
    FOREIGN KEY(user_id)  
        REFERENCES users(user_id),  
    FOREIGN KEY(role_id)  
        REFERENCES roles(role_id)  
);
```

# Alter Table

- Bir tabloya sonradan birincil anahtar eklemek için:

```
ALTER TABLE table_name  
ADD PRIMARY KEY(column_list);
```

```
CREATE TABLE pkdemos(  
    id INT,  
    title VARCHAR(255) NOT NULL  
);
```

```
ALTER TABLE pkdemos  
ADD PRIMARY KEY(id);
```

## KAYNAKLAR

- ▶ MySQL Tutorial - Learn MySQL Fast, Easy and Fun. (2020, March 30). Retrieved from <https://www.mysqltutorial.org>