Create a Database/Schema - Text lecture

Database Server: Service running on a machine (server or desktop), it houses many databases.

Database: Contains all of the persistent information for your applications

Schema: In MySQL, this is synonymous with database and can be used interchangeably, but shouldn't because SQL Server and Oracle treat them differently.
SQL Server/Oracle - Schema is a namespace inside the database with different permissions associated with it.
To clear screen from mysql cli -> \! clear

Open MySQL ->
Refer to the installation videos in section 1 since they differ based on the system you are using (Mac, Windows, AWS Cloud9)
Some commands ->

show databases;

show schemas;

SQL statements end with ";"
If you don't put the ";", MySQL will wait to execute until you ad it in.
To create a new database ->
create database project1;
MySQL is case sensitive.
Don't put spaces in the database name. While possible, it will just cause issues later.. Spaces generally cause annoyances in names of databases, tables, columns and everything.
create schema project1; this statement would accomplish the same thing as the command above to create database

Create another database ->
create database mashrur1;
show databases;

To delete (or drop) the database mashrur1 (warning: this will immediately delete it) ->
drop database mashrur1;

To use a specific database for queries we'll be running and impacting (let's say we want this to be project1):
use project1;
Now when we create and do things like create objects or call objects, database “project1” is assumed.
Note: up arrow recalls previous commands.

Working with tables - Text lecture
- Tables are just like Excel Spreadsheets

- Columns have headers and are for certain types of data like strings, dates, numbers etc.
- Rows are the actual data which all fit inside the column data types.
- Structure matters. Forgetting a comma or a line or a space can break things.

To create a table called people (The capitalization of CREATE TABLE below is to separate out the sql code from the names of the tables, you can also just say create table, capitalization matters in column names):

```
CREATE TABLE people ( PersonID int
                    ,first_name varchar(100)
                    ,last_name  varchar(100)
);
```

To show database in use, type in select database();
show tables;

```
CREATE TABLE people2 ( PersonID INT NOT NULL AUTO_INCREMENT
                     ,FIRST_NAME VARCHAR(100) NULL
                     ,LAST_NAME VARCHAR(100) NULL
                     ,PRIMARY KEY (PersonID));
```

Primary Key is a unique value that each row will have, auto_increment makes it easy by automatically assigning them

If we want to remove a table, use DROP TABLE (or drop table)
show tables;
To add a column, first see the columns that are there by typing in -> show columns from people2;

Notice there are 3 columns;

To add a column called DOB:
alter table people2 add column DOB date NULL;
Similarly, to drop a column, type in:
alter table people2 drop column DOB;
show columns from people2;
To delete tables:
drop table people;
drop table people2;
This will get rid of both tables

Homework exercise: Create two tables - actors and movies
actors will have columns ActorID (as primary key), first_name, last_name and specify primary key
movies will have columns MovieID (as primary key), title, release_year, rating and specify primary key

Inserting rows of data - Text lecture
To add data to tables:

```
INSERT INTO actors ( FIRST_NAME , LAST_NAME )
VALUES ('Ben','Stiller');
```

We'll discuss select further in the next lecture, but to show all data in the table:

```
SELECT *
FROM project1.actors;
```

To insert multiple values into actors table:

```
INSERT INTO project1.actors ( FIRST_NAME , LAST_NAME )
VALUES ('Owen','Wilson')
    ,('Christine','Taylor')
    ,('Will','Ferrell')
    ,('Milla','Jovovich')
    ,('Jerry','Stiller')
    ,('David','Duchovny')
    ,('Jon','Voight')
    ,('Nathan','Graham');
```

Now that we have Zoolander, lets add new actors from Zoolander 2

```
INSERT INTO actors ( FIRST_NAME , LAST_NAME )
```

```
VALUES ('Kristen','Wiig')
      ,('Penélope','Cruz')
      ,('Lenny','Kravitz')
      ,('Macaulay','Culkin')
      ,('Justin','Bieber')
      ,('Cyrus','Arnold');
```

Now let's add data to the movies table:

```
INSERT INTO movies ( Title , ReleaseYear, Rating )
VALUES ('Zoolander',2001,'PG-13')
      ,('Zoolander2',2016,'PG-13')
      ,('Night at the Museum: Secret of the Tomb',2014,'PG')
      ,('Night at the Museum: Battle of the Smithsonian',2009,'PG')
      ,('Night at the Museum',2005,'PG')
      ,('National Treasure',2004,'PG')
      ,('Tropic Thunder',2008,'R');
```

SELECT * FROM movies; -> This will display all data from the movies table

Select - Text lecture

```
SELECT * FROM actors;

SELECT * FROM movies;
```

-> SELECT section lists the results to be returned.
* Means everything
From is where you list tables.
We will add lots of extra details but the SELECT * FROM is the general structure of how we retrieve any and all data from the database.
Listing the columns returns the same results as *

```
SELECT ActorID
      ,FIRST_NAME
      ,LAST_NAME
FROM actors;

SELECT FIRST_NAME
      ,LAST_NAME
FROM actors;
```

Returns same number of rows, but only the columns that were requested (FIRST_NAME AND LAST_NAME)

Concat and substring - Text lecture

CONCAT - Use to add columns together for display

```
SELECT CONCAT(LAST_NAME, ', ', FIRST_NAME)
FROM actors;
```

Returns same number of rows, but with our calculated row.

SQL lets you rename columns for display as well:

```
SELECT CONCAT(LAST_NAME, ', ', FIRST_NAME) AS ACTOR_NAME
FROM actors;
```

To grab certain data from the rows under a column (instead of all the data) you can use substring:

```
SELECT LAST_NAME, SUBSTRING(LAST_NAME,1,3)
FROM actors;
```

A useful query to generate user name from first_name and last_name columns:

```
SELECT FIRST_NAME
     ,LAST_NAME
     ,SUBSTRING(FIRST_NAME,1,1)
     ,CONCAT(SUBSTRING(FIRST_NAME,1,1),LAST_NAME) AS USER_NAME
FROM actors;
```

Select allows you to name columns with spaces using “”, you can create new columns, and calculate columns. You can reuse columns for basically anything.

```
SELECT CONCAT(LAST_NAME, ', ', FIRST_NAME) AS "Actor's Names"
     ,LAST_NAME AS LNAME
     ,FIRST_NAME AS "First Name"
     ,CONCAT(SUBSTRING(FIRST_NAME,1,1),LAST_NAME) AS USER_NAME
     ,'Any Text' AS Anything
     ,1+2       AS Math
FROM actors;
```

Homework challenge - Text version
Phase 1:

Create a table which should have the following columns:
Primary Key (int), Fname, Lname, Game1, Game2, Game3, Game4
Fill in data for the table with your and 3 of your friends names and 4 game scores (remember id for each row, primary key, should be assigned automatically)

Phase 2:
Create a report from the data in the table:
Exclude ID column
- Each column should have a custom header/title, atleast 1 should have a space in it
- A column with player initials
- A column with last name, first name separated by comma
- Return each game's score
- Column with total score of all four games for each player
- Column with average score for each player
IMPORTANT: Please post your code to the Q & A section for this lecture (or the prior video lecture)

Solution - Text

```
CREATE TABLE players1 (BowlerID INT NOT NULL AUTO_INCREMENT

                    ,FNAME varchar(50) DEFAULT NULL
                    ,LNAME varchar(50) DEFAULT NULL
                    ,Game1 int DEFAULT NULL
                    ,Game2 int DEFAULT NULL
                    ,Game3 int DEFAULT NULL
                    ,Game4 int DEFAULT NULL
                    ,PRIMARY KEY (BowlerID));
INSERT INTO players1 (FNAME, LNAME,Game1,Game2,Game3,Game4)
VALUES ('Mashrur', 'Hossain',121,87,115,124)
     ,('Matt', 'Berstein',111,99,135,105)
     ,('Anastasia', 'Ivanov',75,99,125,141)
     ,('Mark', 'Futre',115,128,101,84);
```

```
SELECT *
FROM players1;
SELECT CONCAT(LNAME,', ',FNAME) AS Player
    ,CONCAT(SUBSTRING(FNAME,1,1),SUBSTRING(LNAME,1,1)) AS Initials
    ,GAME1 AS G1
    ,GAME2 AS G2
    ,GAME3 AS G3
    ,GAME4 AS G4
    ,GAME1 + GAME2 + GAME3 + GAME4 AS "Tournament Total"
    ,(GAME1 + GAME2 + GAME3 + GAME4)/4 AS "Tournament Average"
FROM players1;
```

Alternate method of running queries - script - Text version

To run a sql script, create a file (example: test1.sql) in your working directory and save it. Add all the sql code you want to run in there.

Start a SQL cli session and use the source keyword to execute it. If you wanted to run test1.sql type in:
source test1.sql

Limit, order by and distinct - Text lecture

Large datasets where you don't want to tax the system returning millions of rows use limit:

```
SELECT *
FROM actors
LIMIT 10;
```

This query above will return the first 10 results from the actors table instead of all the rows.

Order by can be used to sort the data that is returned in the query result:

```
SELECT *
FROM actors
ORDER BY 3;
```

This query above will return the results sorted by the 3rd column of the table in ascending order by default

Which is the same as
SELECT *
FROM actors
ORDER BY LAST_NAME ASC;
The ASC is for ascending which is implied. The opposite is descending DESC

SELECT *
FROM actors
ORDER BY 3 DESC;
We can also sort by multiple columns at the same time and by name
SELECT *
FROM actors
ORDER BY LAST_NAME ASC, FIRST_NAME DESC;
Notice the order the Jerry and Ben are in.
And you can combine LIMIT and ORDER BY
SELECT *
FROM project1.actors
ORDER BY LAST_NAME ASC
	,FIRST_NAME ASC
LIMIT 10;
DISTINCT

Let's say you want to know all of the ratings for the movies
SELECT RATING
FROM movies;
But don’t like the duplicate rows and want to only display unique ratings.
SELECT DISTINCT RATING
FROM movies;
Distinct only works when the whole row returned is unique. In the query below, because each movie has a unique ID, none of the rows are removed.
SELECT DISTINCT MovieID
		,RATING
FROM movies;

Count, like and group by - Text lecture

Count can be used as follows:

```
SELECT COUNT(*) AS cnt
FROM movies;
```

This will return the number of movies in the table

Try the following to get the number of movies for each rating and you'll get incorrect results:

```
SELECT RATING
    ,COUNT(*) AS cnt
FROM movies;
```

*Incorrect Results

You can use GROUP BY (column name) to get correct results in such cases:

```
SELECT RATING
    ,COUNT(*) AS MOVIES
FROM movies
GROUP BY RATING;
```

Like is used all the time for search when an entire description is not known, it's used with a % -> LIKE %

```
SELECT *
FROM movies
WHERE TITLE like 'Night at the Museum%';
```

Name the return columns with spaces using:

```
SELECT COUNT(*) AS "Night at the Museum Movies"
FROM movies
WHERE TITLE like 'Night at the Museum%';
```

You can put % before the word as well so like '%Museum%' and it will return all results that have museum in the title

Min and max - Text lecture

```
SELECT MAX(ReleaseYear)

FROM movies;
```

They work exactly how you would expect.

```
SELECT Title
    ,MAX(ReleaseYear)
```

```
    ,RATING
FROM movies;
*Incorrect results
SELECT MIN(ReleaseYear)
    ,RATING
FROM movies
WHERE Title like 'Night at the Museum%'
GROUP BY Title
     ,RATING;
SELECT RATING
    ,MIN(ReleaseYear)
FROM movies
WHERE Title like 'Night at the Museum%'
GROUP BY RATING;
SELECT RATING
    ,MIN(ReleaseYear)
FROM movies
GROUP BY RATING;
SELECT RATING
    ,MAX(ReleaseYear)
FROM movies
GROUP BY RATING;
```

MySQL is more “flexible” than other SQL versions. It doesn’t require the GROUP BY but it should. It is likely you will get unhelpful or incorrect results when you do not use a group by.

All result sets should either be part of an aggregate function, or in the Group By.

Code challenge project - Text version

```
CREATE TABLE bowlResults (BowlResultID INT NOT NULL
AUTO_INCREMENT

                ,FNAME varchar(50) DEFAULT NULL
                ,LNAME varchar(50) DEFAULT NULL
                ,Game_Num int DEFAULT NULL
```

```
                    ,Game_Score int DEFAULT NULL
                    ,PRIMARY KEY (BowlResultID));
INSERT INTO bowlResults (FNAME, LNAME,Game_Num, Game_Score)
VALUES ('Mashrur', 'Hossain',1,121)
     ,('Mashrur', 'Hossain',2,87)
     ,('Mashrur', 'Hossain',3,115)
     ,('Mashrur', 'Hossain',4,124)
     ,('Matt', 'Berstein',1,111)
     ,('Matt', 'Berstein',2,99)
     ,('Matt', 'Berstein',3,135)
     ,('Matt', 'Berstein',4,105)
     ,('Anastasia', 'Ivanov',1,75)
     ,('Anastasia', 'Ivanov',2,99)
     ,('Anastasia', 'Ivanov',3,125)
     ,('Anastasia', 'Ivanov',4,141)
     ,('Mark', 'Futre',1,115)
     ,('Mark', 'Futre',2,128)
     ,('Mark', 'Futre',3,101)
     ,('Mark', 'Futre',4,84);
SELECT *
FROM bowlResults;
SELECT CONCAT(FNAME,' ',LNAME)  AS Player
     ,SUM(Game_Score)          AS "Tournament Total"
     ,SUM(Game_Score)/COUNT(*) AS "Tournament Ave"
FROM  bowlResults
GROUP BY FNAME, LNAME
ORDER BY  SUM(Game_Score)/COUNT(*) DESC
LIMIT 3;
SELECT CONCAT(FNAME,' ',LNAME)  AS Player
     ,MAX(Game_Score)          AS "Best Game"
FROM  bowlResults
GROUP BY FNAME, LNAME
ORDER BY  MAX(Game_Score) DESC;
SELECT DISTINCT FNAME AS "First Name"
              ,LNAME AS "Last Name"
FROM  bowlResults;
```