SQL Homework 2

SQL, DDL and DDM

SQL Exercises

- Selecting Data
- Filtering Data
- Counting Rows
- Creating Calculated Fields
- Checking Data Exists
- Multiple Selections

Initializing Databases

- Before beginning any set of selection queries, make sure
  - The database you want to use is created
  - The tables are created
  - Data exists in the tables
  - You are using the database

Using the Database

- Using the database is simple. Simply type the following statement (make sure you have a semicolon)

```
USE PersonalTrainer;
```

Selecting Data

- You can select all the data from a table by typing the following command (make sure the table exists)

```
SELECT * FROM [tablename]
```

```sql
-- Activity 1
SELECT * FROM Exercise;

-- Activity 2
SELECT * FROM Client;
```

Filtering Data

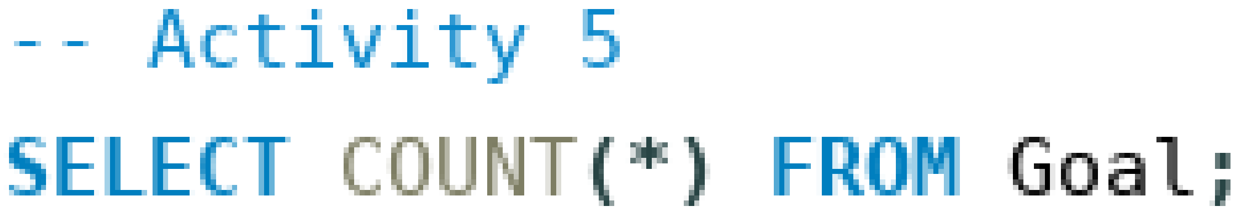
- Rarely do you ever want the entire dataset when searching through data
  - To filter down data, you can use WHERE to create conditions

```sql
-- Activity 3
SELECT * FROM Client WHERE City = "Metairie";

-- Activity 4
SELECT * FROM Client WHERE ClientId = "818u7faf-7b4b-48a2-bf12-7a26c92de20c";
```

Counting Rows

- Special functions such as Count exist to help with easy activities like counting the number of rows
  - Be smart and let the computer do the work for you



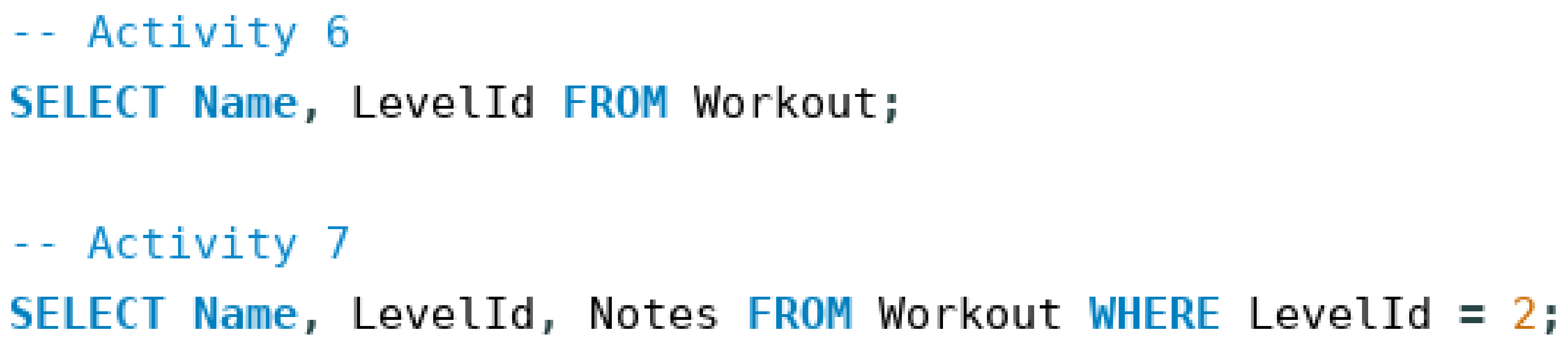
```
-- Activity 5
SELECT COUNT(*) FROM Goal;
```

Other Functions

- AVG
- MAX
- MIN
- SUM

Selecting Multiple Columns

- You can select multiple columns and also apply conditional statements to them

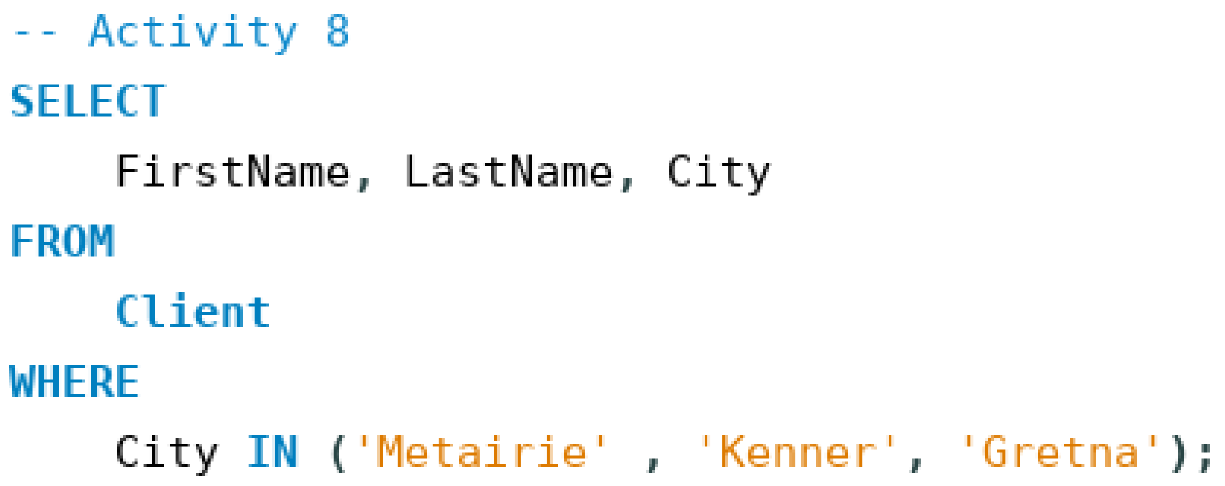


```sql
-- Activity 6
SELECT Name, LevelId FROM Workout;

-- Activity 7
SELECT Name, LevelId, Notes FROM Workout WHERE LevelId = 2;
```

Filtering Based on Multiple Values

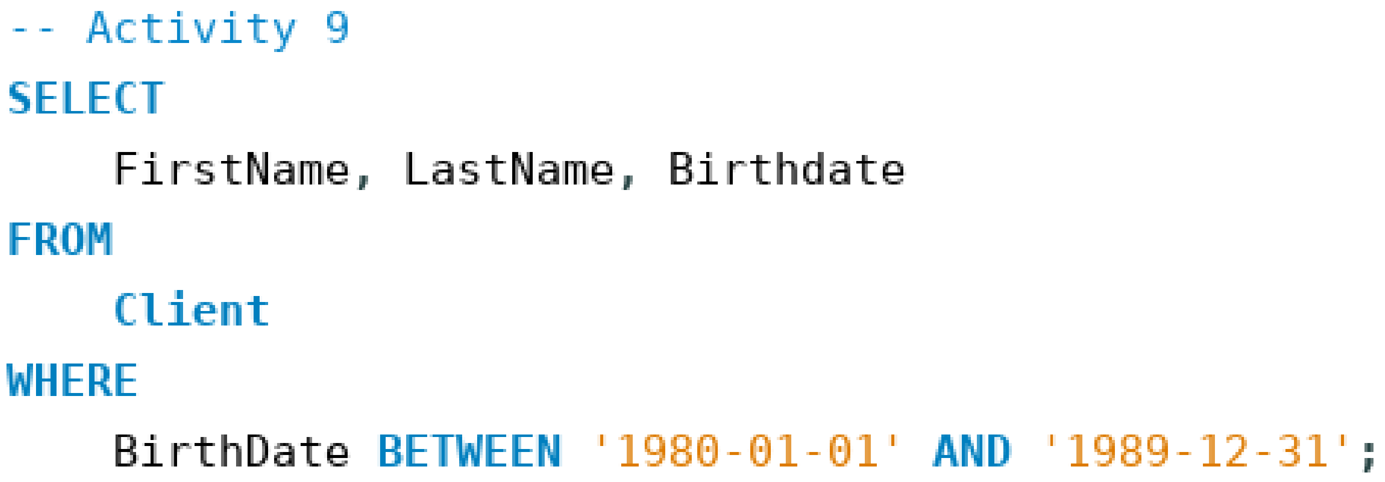
- You can pass multiple possible values using the IN statement



```sql
-- Activity 8
SELECT
    FirstName, LastName, City
FROM
    Client
WHERE
    City IN ('Metairie' , 'Kenner', 'Gretna');
```

Filtering Based on Ranges

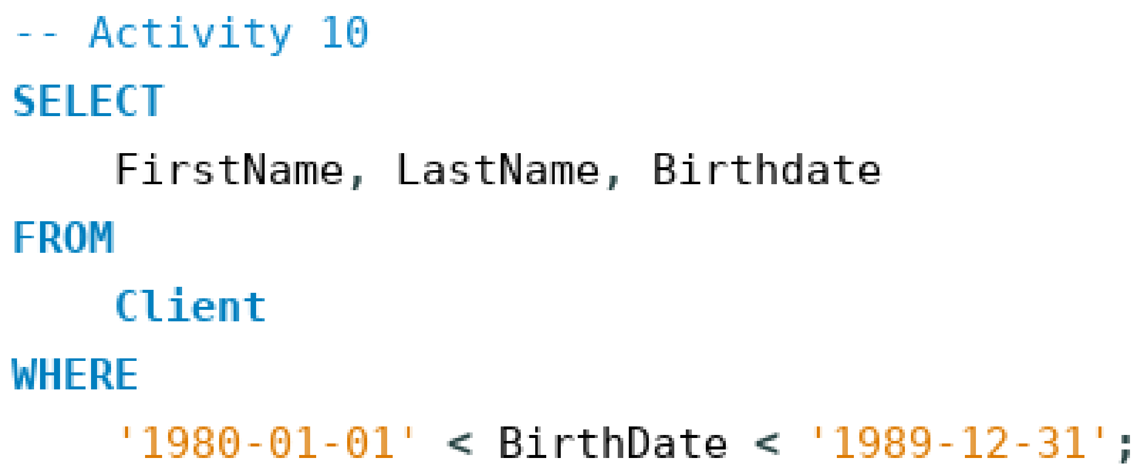
- For numeric values such as INT or Dates, you can filter based on ranges using BETWEEN



```sql
-- Activity 9
SELECT
    FirstName, LastName, Birthdate
FROM
    Client
WHERE
    BirthDate BETWEEN '1980-01-01' AND '1989-12-31';
```

Filtering Based on Ranges

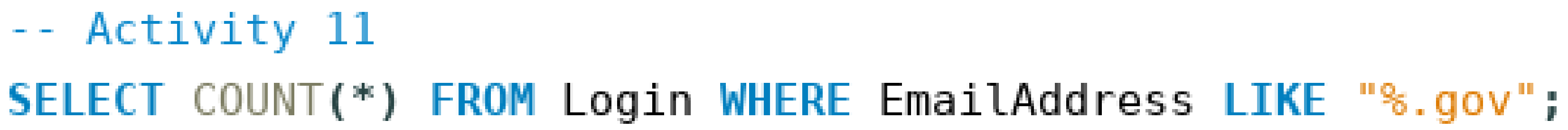
- You can use the lesser than ‘<’ or greater than ‘>’ symbols as well to indicate a range



```sql
-- Activity 10
SELECT
    FirstName, LastName, Birthdate
FROM
    Client
WHERE
    '1980-01-01' < BirthDate < '1989-12-31';
```

Regular Expressions

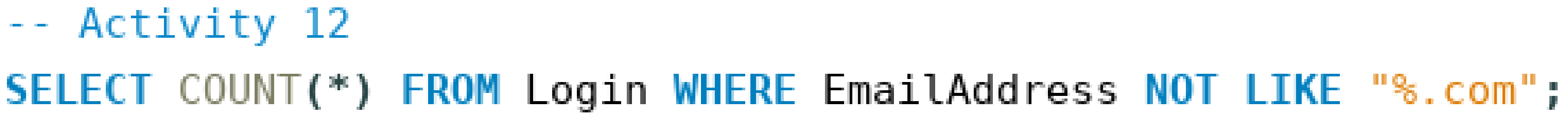
- You can use regular expressions to filter the data as well using the LIKE statement



```sql
-- Activity 11
SELECT COUNT(*) FROM Login WHERE EmailAddress LIKE "%.gov";
```

NOT Statements

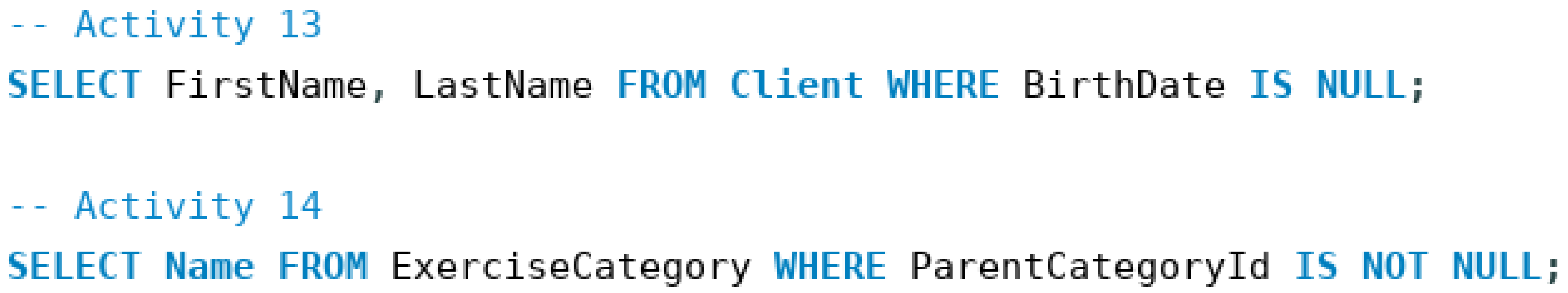
- Using the NOT statement excludes any conditions that you have set



```sql
-- Activity 12
SELECT COUNT(*) FROM Login WHERE EmailAddress NOT LIKE "%.com";
```

Filtering NULL Values

- NULL values are special values in SQL that require you to check if a value is or is not NULL, rather than using ‘= NULL’ or similar checks

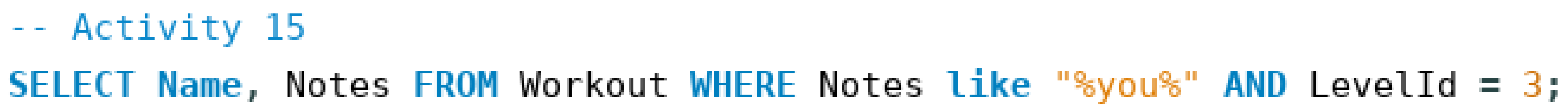


```
-- Activity 13
SELECT FirstName, LastName FROM Client WHERE BirthDate IS NULL;

-- Activity 14
SELECT Name FROM ExerciseCategory WHERE ParentCategoryId IS NOT NULL;
```

Combining Conditions

- You can combine multiple conditions in a single WHERE statement for more powerful and precise filtering



```
-- Activity 15
SELECT Name, Notes FROM Workout WHERE Notes like "%you%" AND LevelId = 3;
```

```sql
-- Activity 16
SELECT
    FirstName, LastName, City
FROM
    Client
WHERE
    (LastName LIKE 'L%' OR LastName LIKE 'M%'
        OR LastName LIKE 'N%')
        AND City = 'LaPlace';
```

Calculated Columns

- You can calculate existing fields and create new fields that can be referred to later on

```sql
-- Activity 17
SELECT
    InvoiceId,
    Description,
    Price,
    Quantity,
    ServiceDate,
    (Price*Quantity) AS Line_Item_Total
FROM
    InvoiceLineItem
HAVING Line_Item_Total BETWEEN 15 AND 25;
```

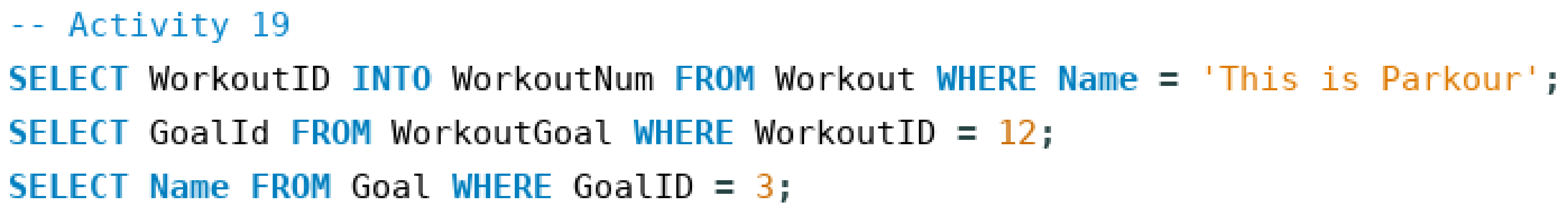
Calculated Columns

- However, using these calculated fields as filter conditions requires the ‘HAVING’ statement rather than WHERE

```
-- Activity 17
SELECT
    InvoiceId,
    Description,
    Price,
    Quantity,
    ServiceDate,
    (Price*Quantity) AS Line_Item_Total
FROM
    InvoiceLineItem
HAVING Line_Item_Total BETWEEN 15 AND 25;
```

Conditionally Selecting

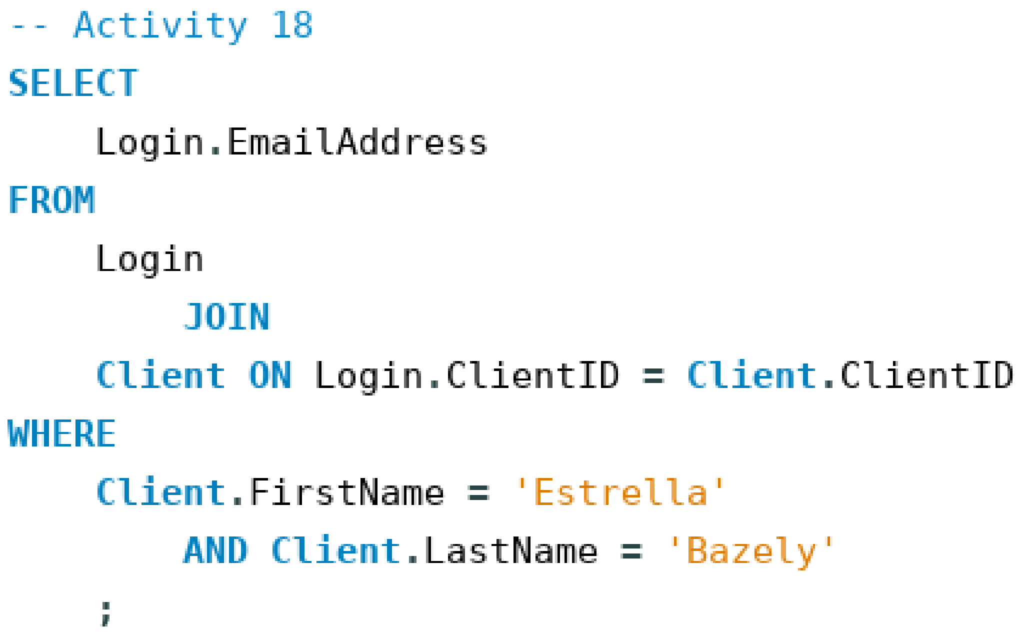
- To select based on the output of another table (especially if filtered), you can use two methods
  - Manual referencing of the values (or you can store output INTO variables that can be later referenced)



```
-- Activity 19
SELECT WorkoutID INTO WorkoutNum FROM Workout WHERE Name = 'This is Parkour';
SELECT GoalId FROM WorkoutGoal WHERE WorkoutID = 12;
SELECT Name FROM Goal WHERE GoalID = 3;
```

Conditionally Selecting

- Or using a JOIN statement



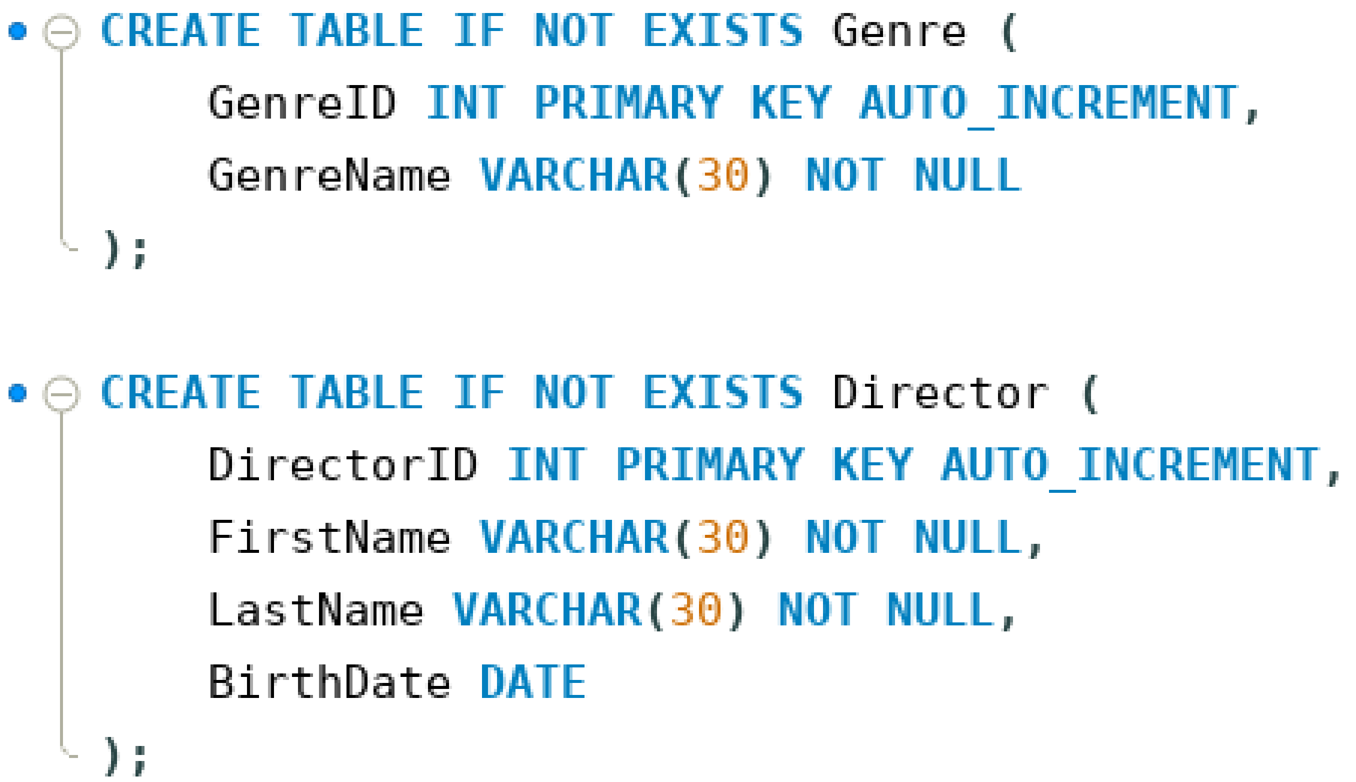
```
-- Activity 18
SELECT
    Login.EmailAddress
FROM
    Login
        JOIN
    Client ON Login.ClientID = Client.ClientID
WHERE
    Client.FirstName = 'Estrella'
        AND Client.LastName = 'Bazely'
    ;
```

DDL

- Data Structuring
- Setting Constraints
- Altering Tables

Data Structuring

- In SQL, it’s important to define
  - What type of data a field is
  - Whether it is required (NOT NULL)
  - If it can auto increment (such as for keys)



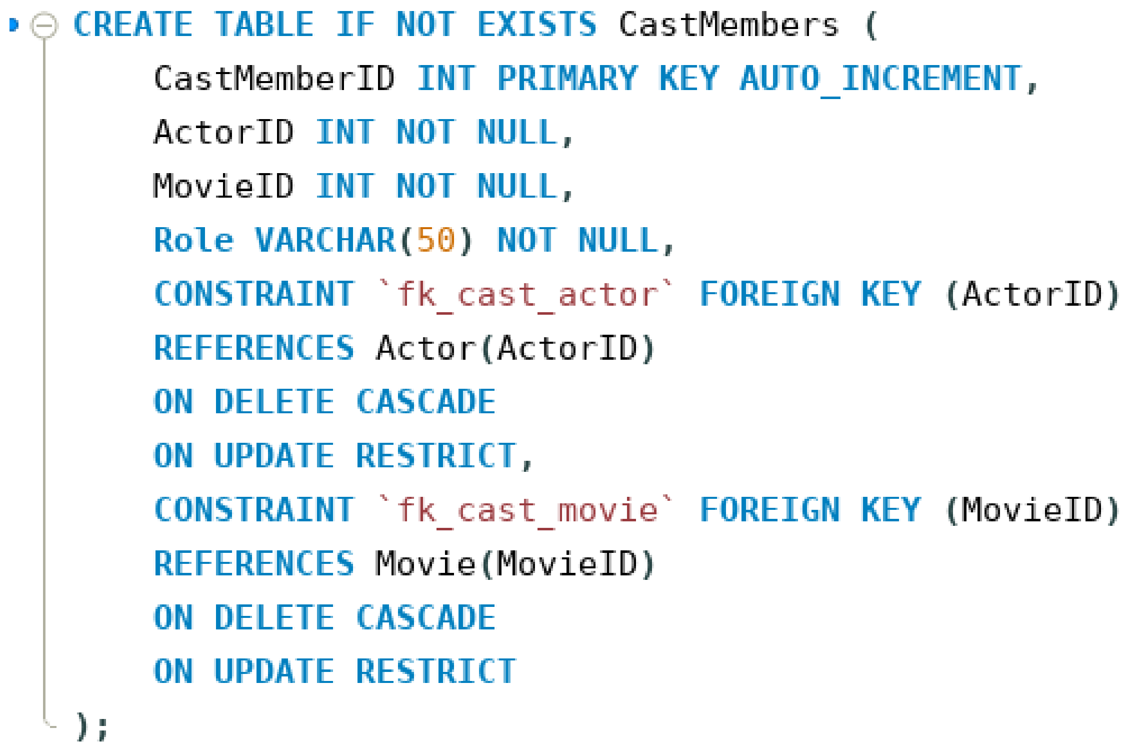
```sql
CREATE TABLE IF NOT EXISTS Genre (
    GenreID INT PRIMARY KEY AUTO_INCREMENT,
    GenreName VARCHAR(30) NOT NULL
);

CREATE TABLE IF NOT EXISTS Director (
    DirectorID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(30) NOT NULL,
    LastName VARCHAR(30) NOT NULL,
    BirthDate DATE
);
```

Setting Constraints

- When using foreign keys, it’s good to have names so that they can be referenced, and setting constraints helps to determine what happens when data is altered as well

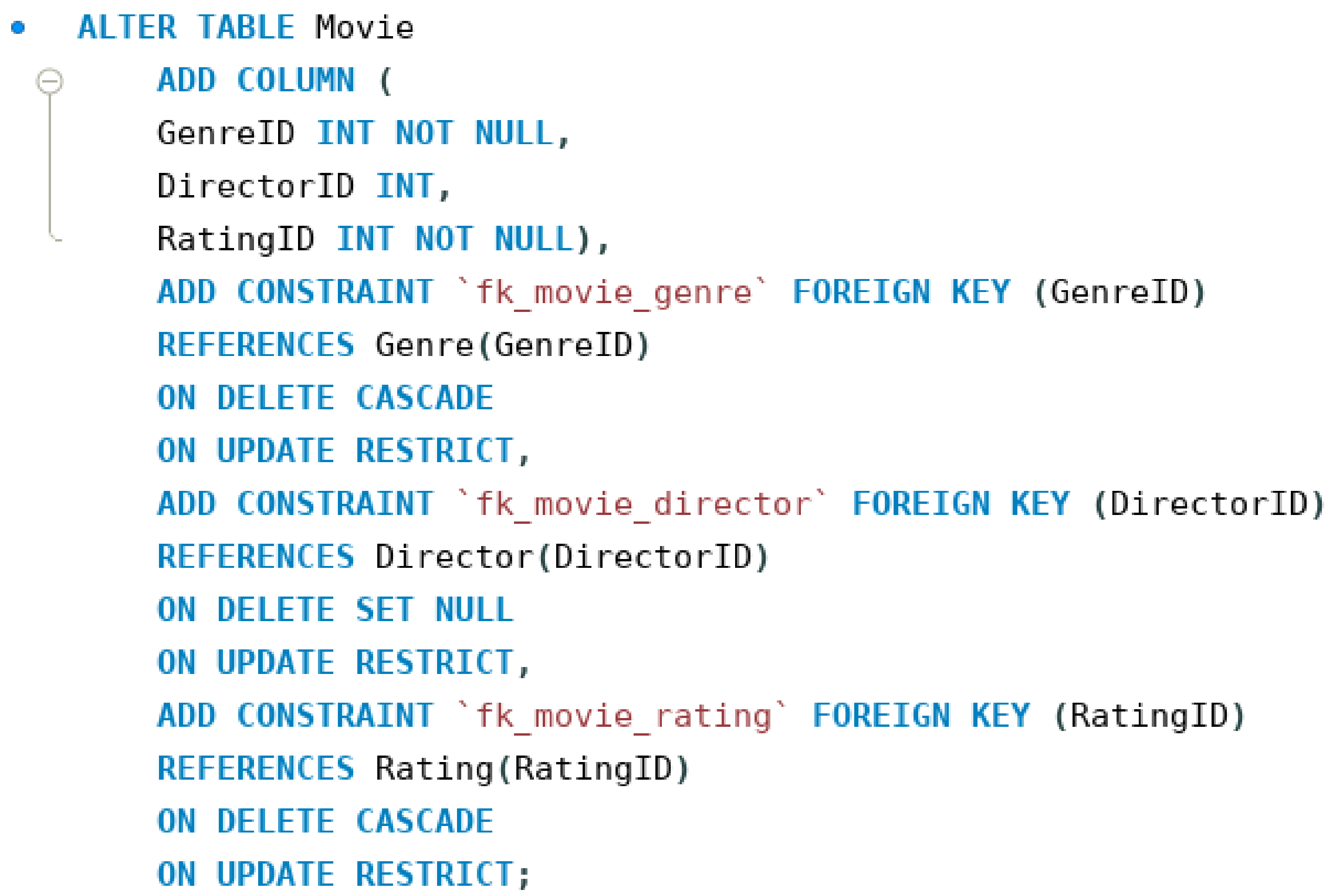
Setting Constraints



```
CREATE TABLE IF NOT EXISTS CastMembers (
    CastMemberID INT PRIMARY KEY AUTO_INCREMENT,
    ActorID INT NOT NULL,
    MovieID INT NOT NULL,
    Role VARCHAR(50) NOT NULL,
    CONSTRAINT `fk_cast_actor` FOREIGN KEY (ActorID)
    REFERENCES Actor(ActorID)
    ON DELETE CASCADE
    ON UPDATE RESTRICT,
    CONSTRAINT `fk_cast_movie` FOREIGN KEY (MovieID)
    REFERENCES Movie(MovieID)
    ON DELETE CASCADE
    ON UPDATE RESTRICT
);
```

Altering Tables

- After creating tables, if you want to add columns or foreign keys, you need to do so through the ALTER statement



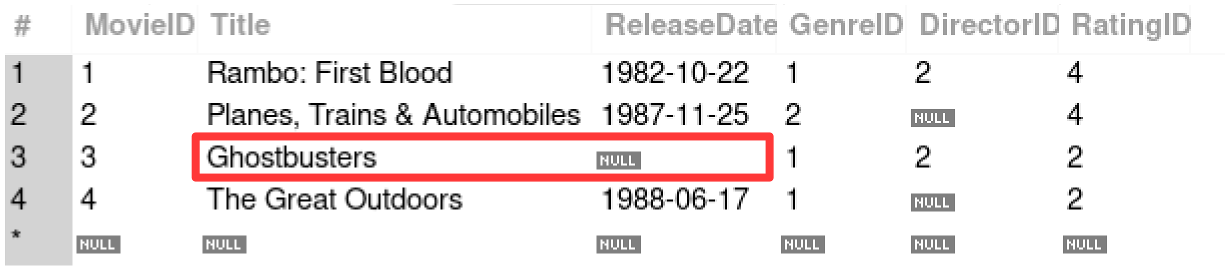
```
ALTER TABLE Movie
    ADD COLUMN (
    GenreID INT NOT NULL,
    DirectorID INT,
    RatingID INT NOT NULL),
    ADD CONSTRAINT `fk_movie_genre` FOREIGN KEY (GenreID)
    REFERENCES Genre(GenreID)
    ON DELETE CASCADE
    ON UPDATE RESTRICT,
    ADD CONSTRAINT `fk_movie_director` FOREIGN KEY (DirectorID)
    REFERENCES Director(DirectorID)
    ON DELETE SET NULL
    ON UPDATE RESTRICT,
    ADD CONSTRAINT `fk_movie_rating` FOREIGN KEY (RatingID)
    REFERENCES Rating(RatingID)
    ON DELETE CASCADE
    ON UPDATE RESTRICT;
```

DDM

- Changing Data
- Deleting Data
- Adding Data

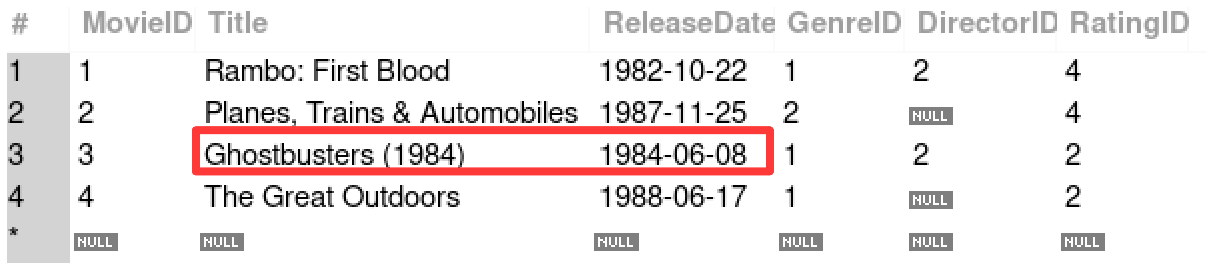
Changing Data

- Often, you may wish to change data in a given table
- You can do this through using the UPDATE and SET statements



#	MovieID	Title	ReleaseDate	GenreID	DirectorID	RatingID
1	1	Rambo: First Blood	1982-10-22	1	2	4
2	2	Planes, Trains & Automobiles	1987-11-25	2	NULL	4
3	3	Ghostbusters	NULL	1	2	2
4	4	The Great Outdoors	1988-06-17	1	NULL	2
*	NULL	NULL	NULL	NULL	NULL	NULL

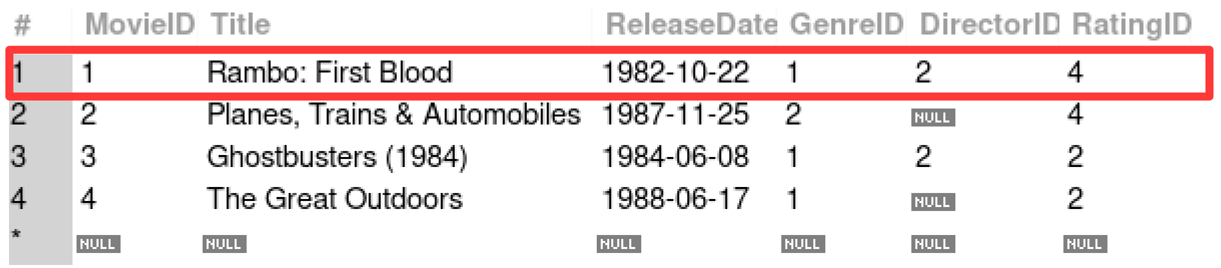
```
UPDATE Movie SET
    Title = "Ghostbusters (1984)",
    ReleaseDate = STR_TO_DATE("6/8/1984", "%m/%d/%Y")
WHERE
    Title = "Ghostbusters";
```



#	MovieID	Title	ReleaseDate	GenreID	DirectorID	RatingID
1	1	Rambo: First Blood	1982-10-22	1	2	4
2	2	Planes, Trains & Automobiles	1987-11-25	2	NULL	4
3	3	Ghostbusters (1984)	1984-06-08	1	2	2
4	4	The Great Outdoors	1988-06-17	1	NULL	2
*	NULL	NULL	NULL	NULL	NULL	NULL

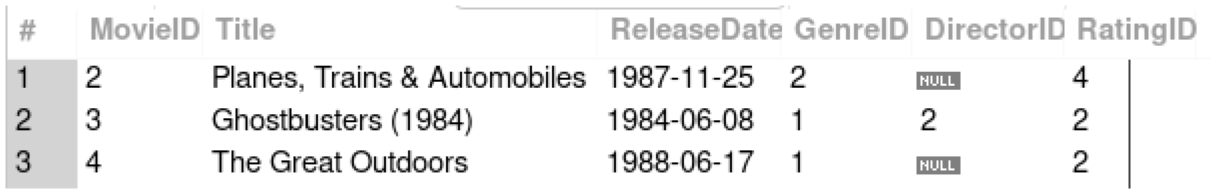
Deleting Data

- You may also want to delete data from a table
- You can do this with the DELETE statement



#	MovieID	Title	ReleaseDate	GenreID	DirectorID	RatingID
1	1	Rambo: First Blood	1982-10-22	1	2	4
2	2	Planes, Trains & Automobiles	1987-11-25	2	NULL	4
3	3	Ghostbusters (1984)	1984-06-08	1	2	2
4	4	The Great Outdoors	1988-06-17	1	NULL	2
*	NULL	NULL	NULL	NULL	NULL	NULL

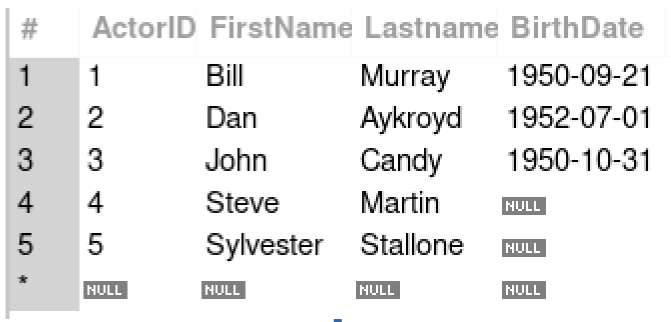
```sql
DELETE FROM Movie
WHERE Title = "Rambo: First Blood";
```



#	MovieID	Title	ReleaseDate	GenreID	DirectorID	RatingID
1	2	Planes, Trains & Automobiles	1987-11-25	2	NULL	4
2	3	Ghostbusters (1984)	1984-06-08	1	2	2
3	4	The Great Outdoors	1988-06-17	1	NULL	2

Adding Data

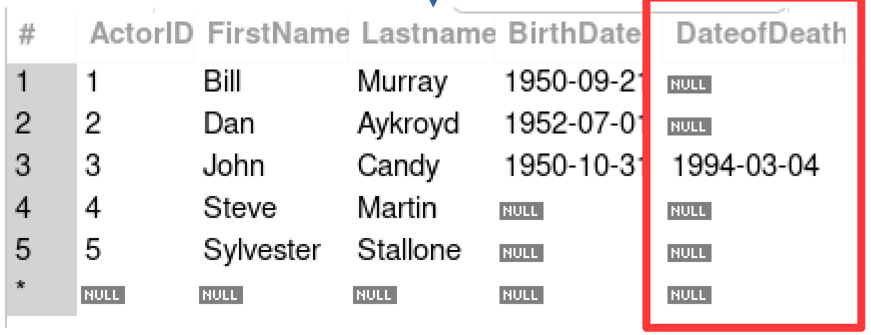
- You may also want to add data, like new columns rather than simply inserting into existing data
- You can do this through a combination of adding a new column and setting values



#	ActorID	FirstName	Lastname	BirthDate
1	1	Bill	Murray	1950-09-21
2	2	Dan	Aykroyd	1952-07-01
3	3	John	Candy	1950-10-31
4	4	Steve	Martin	NULL
5	5	Sylvester	Stallone	NULL
*	NULL	NULL	NULL	NULL

```
ALTER TABLE Actor
    ADD COLUMN (DateofDeath Date);

UPDATE Actor SET
    DateofDeath = STR_TO_DATE("3/4/1994", "%m/%d/%Y")
WHERE
    FirstName = "John" AND Lastname = "Candy";
```



#	ActorID	FirstName	Lastname	BirthDate	DateofDeath
1	1	Bill	Murray	1950-09-2	NULL
2	2	Dan	Aykroyd	1952-07-0	NULL
3	3	John	Candy	1950-10-3	1994-03-04
4	4	Steve	Martin	NULL	NULL
5	5	Sylvester	Stallone	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL

Questions?