SQL Coding

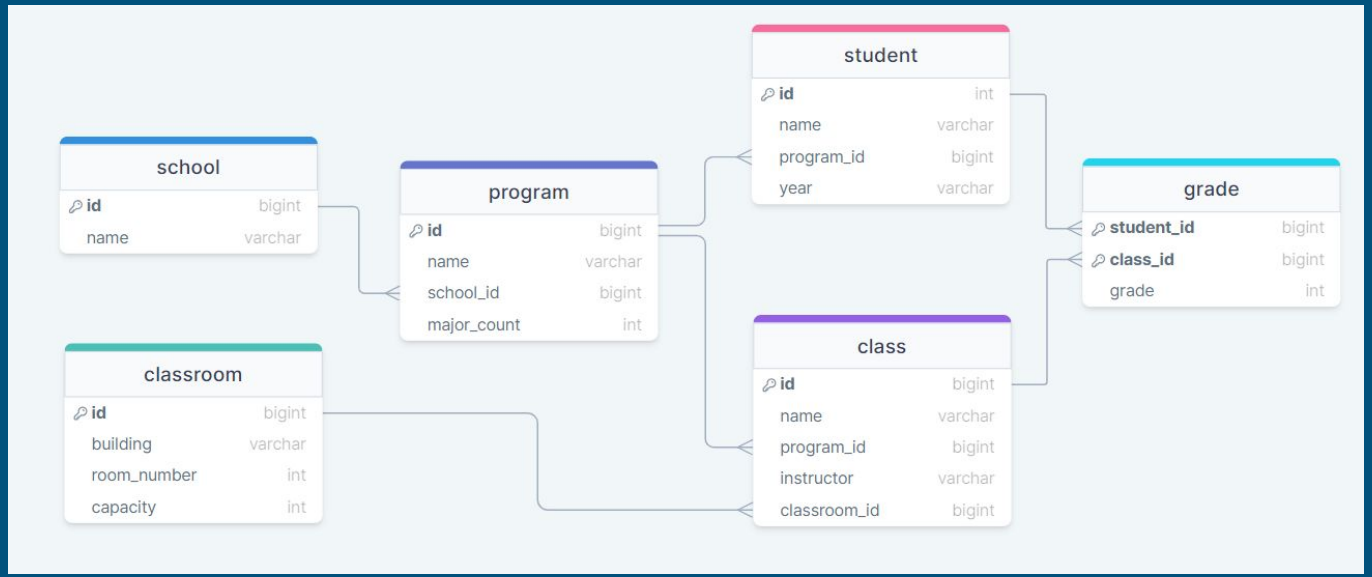
CRUD: Creating, Reading, Updating, and Deleting

What will be covered today

- Todo
  - How to make a mysql database- CREATE DATABASE test;
  - VSCODE setup- sql tools and sql tools mysql- both by Matheus Teixeira
  - Make one table and explore basic commands- CRUD
    - Create student table with no constraints, insert some data- show how some weird things can happen
    - delete/drop the table
    - Make it again and this time add constraints
      - Auto_incremement
      - Example with not null, unique, and default
    - Update year where year is freshman
    - Talk briefly about boolean operators
    - Select statements with where statements as well
    - Drop table again so we can start fresh
  - Make berry college database from my example
    - Talk about primary keys and foreign keys
    - For first table- will not be able to give it a foreign key cus no reference- show this error
    - Make other tables with foreign keys
    - Alter command to add foreign key back

Lets review

- What is a **Database Schema?**



Lets now make it! With actual SQL code!

1. Create a Database via the mySQL command prompt
2. Set up VSCode to connect to our mySQL Database
3. Write SQL code to make tables and explore them
    a. CRUD
4. Form relationships between tables
5. Take advantage of relational databases to get data from multiple tables at once!

Create a database via mySQL command line

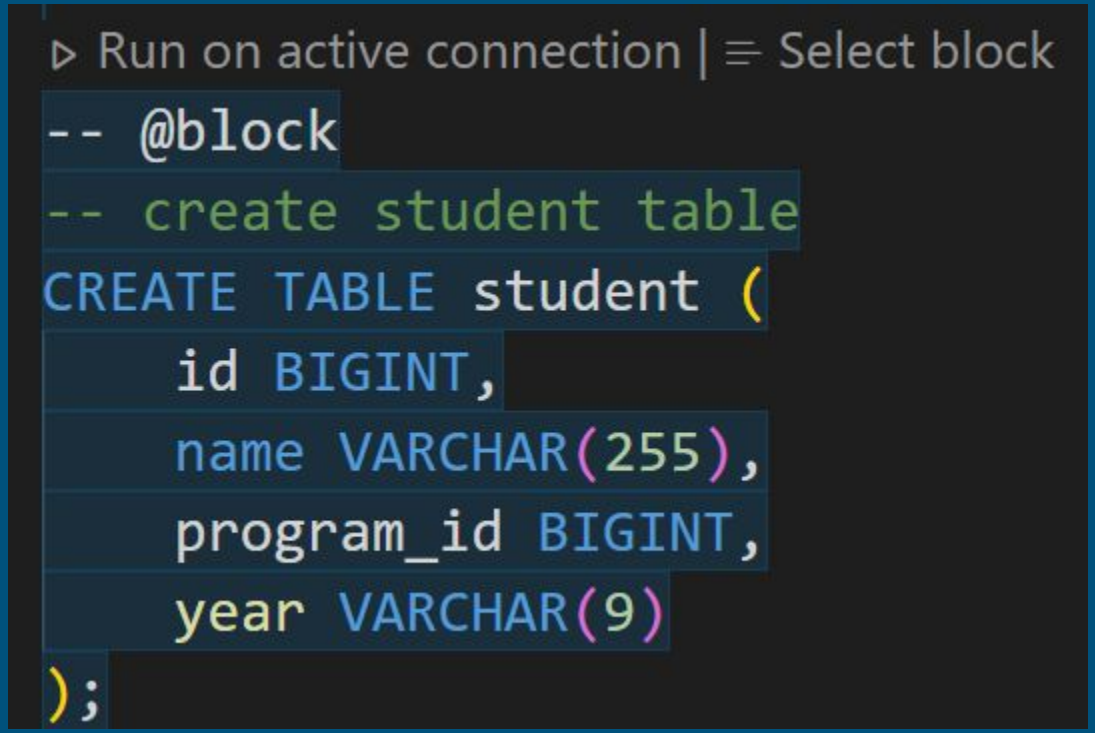
Do this:

```
mysql> CREATE DATABASE testDB;
```

Lets talk about CRUD

- CRUD
  - Create
  - Read
  - Update
  - Delete

Creating Tables



Exploring tables

Show what tables exist-

```
SHOW tables
```

Get cols from “student” table

```
DESCRIBE student
```

Inserting Data into a table

```
INSERT INTO student (id, name, program_id, year ) VALUES
    (1, "Sean", 1, "sophomore"),
    (2, "Ben", 3, "junior" );
```

How to query data

```
SELECT * FROM student
```

- * means “all”
- Can put col name as well

How to delete a table

```
DROP TABLE student
```

Talk about constraints

- `NOT NULL` - Ensures that a column cannot have a NULL value
- `UNIQUE` - Ensures that all values in a column are different
- `PRIMARY KEY` - A combination of a `NOT NULL` and `UNIQUE`. Uniquely identifies each row in a table
- `FOREIGN KEY` - Prevents actions that would destroy links between tables
- `CHECK` - Ensures that the values in a column satisfies a specific condition
- `DEFAULT` - Sets a default value for a column if no value is specified
- `CREATE INDEX` - Used to create and retrieve data from the database very quickly

Talk about constraints

- Check out how applying a unique constraint limits what you can place into cols
  - Make name col unique
- Check out auto_incrememt

How to change

```
UPDATE student
SET year = "senior"
WHERE year = "junior"
```

Conditionals

```
SELECT * FROM student
WHERE name = "Sean"
```

Deleting Rows

```
DELETE FROM student
WHERE name = "Ben";
```

Now lets make a relational database!

- How do we relate different tables?

Now lets make a relational database!

- How do we relate different tables?
  - We connect a col’s foreign key to a col in the primary key of another table!

Talk about Primary Keys

```
CREATE TABLE program (
    id BIGINT AUTO_INCREMENT,
    name VARCHAR(255),
    school_id BIGINT,
    major_count int,
    PRIMARY KEY (id)
);
```

What about Foreign Key?

```
CREATE TABLE student (
    id BIGINT AUTO_INCREMENT,
    name VARCHAR(255),
    program_id BIGINT,
    year VARCHAR(255),
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES grade(student_id) ON DELETE CASCADE,
    FOREIGN KEY (program_id) REFERENCES program(id) ON DELETE SET NULL
);
```

You cannot make a foreign key to table that doesnt exist!

- How do you handle this?
- Make tables first and then “alter” the table so it has a foreign key constraint

Altering an existing table to have a foreign key

```
ALTER TABLE program
ADD FOREIGN KEY (school_id)
REFERENCES school(id)
ON DELETE SET NULL
```

If child table exists, we can make foreign key up front

```
CREATE TABLE student (
    id BIGINT AUTO_INCREMENT,
    name VARCHAR(255),
    program_id BIGINT,
    year VARCHAR(255),
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES grade(student_id) ON DELETE CASCADE,
    FOREIGN KEY (program_id) REFERENCES program(id) ON DELETE SET NULL
);
```

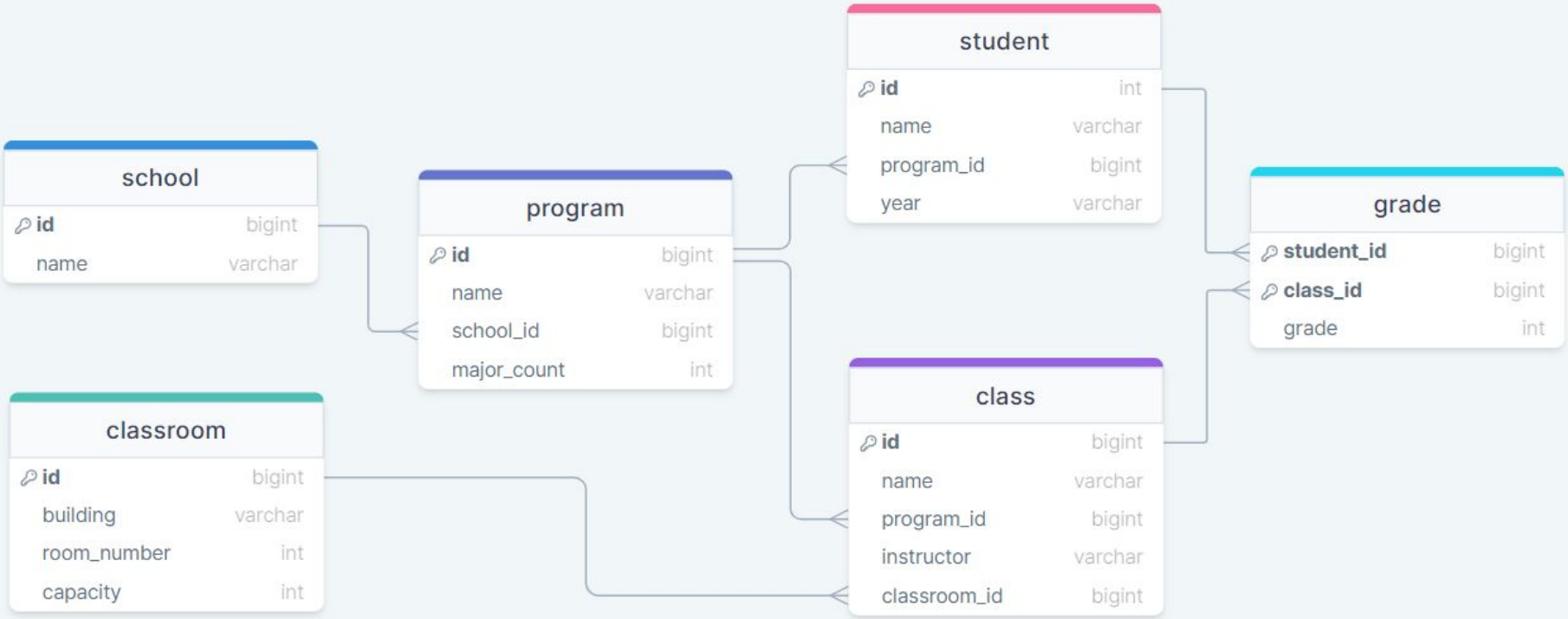
Go ahead and make all tables with all relations

- Follow along with instructor

Time to insert data!

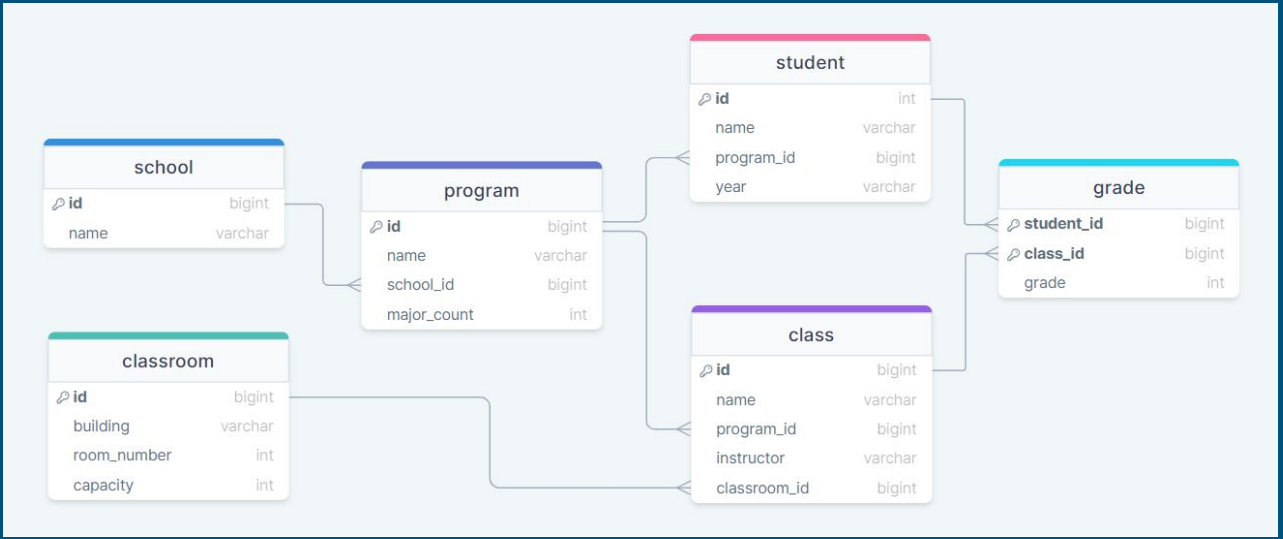
- Order you insert matters!
- SQL will maintain relations
- Ex:
  - Can you have a class without a classroom?
    - Must have data in classroom table first before putting data in class

Here is schema:



Here is schema:

What do you notice about the relations and what tables that have to go first?



One-to-many

- If you have one-to-many relation, you must insert data into the table with “one” before you insert into the table with “many”
- One-to-one- doesn’t matter what order you do

Finish inserting data

- Follow along with instructor

Lets now write some queries!

- What would you all like to ask?

Lets talk about JOINS

- A JOIN gets data from multiple tables at once
  - It is only possible between tables that have a relation between a foreign key and a primary key

Ex:

```
SELECT NAME FROM grade JOIN student
ON grade.student_id = student.id
WHERE grade.grade > 90
```

- Grade is table 1
- Student is table 2
- The student_id col in grade table is foreign key to student table’s id col
- Where statement is used like usual

Lets try some joins!

Daily Assignment