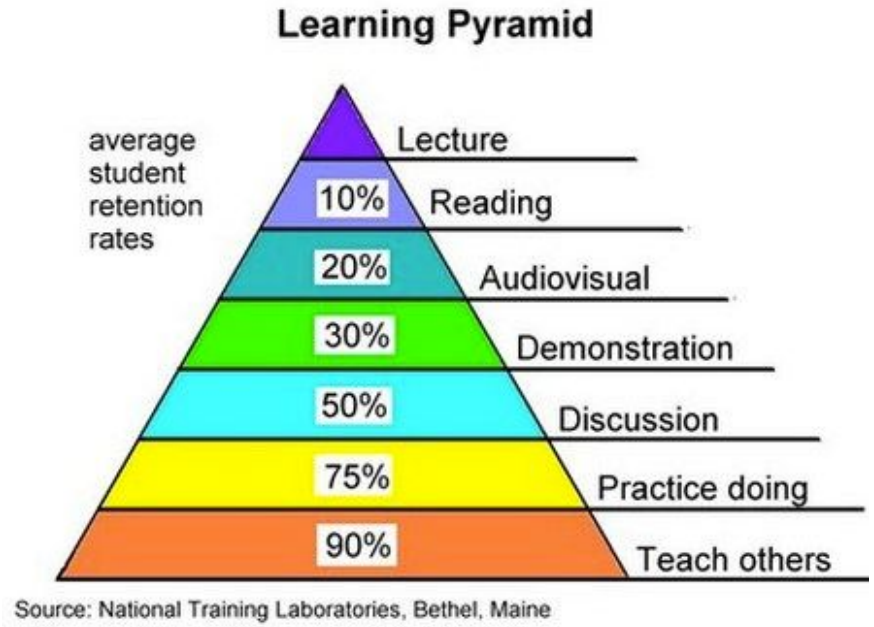
SQL Basics (Part 1)

Kian

Before you start

- This is a simple tutorial with some hands-on exercises.
- You **won't learn** just by reading. You need to take some notes. Grab a notebook and a pencil. If you are comfortable with markdowns, you can download this note taking app call Boostnote.

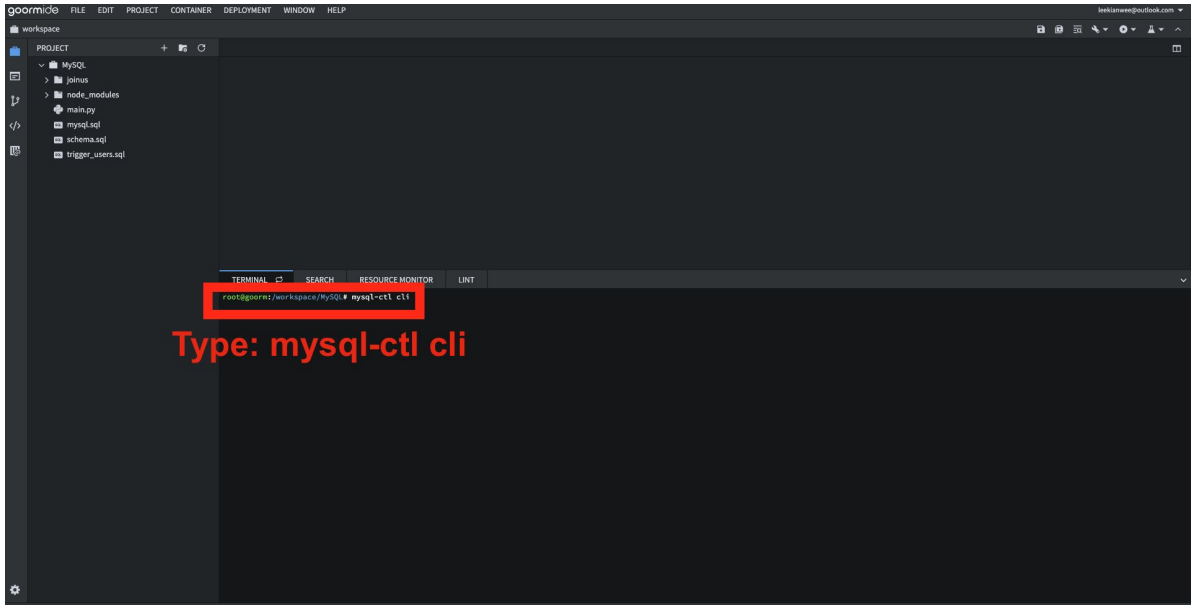


SQL vs MySQL

- In simple terms, SQL is a language we used to talk to our database.
  - Like find all users, add users with name ‘Prison Mike’.
- MySQL, similar to Oracle, MariaDB and SQLite is a Relational Database Management System (RDMS).
  - It’s an open source software based on the SQL language
  - To use SQL, you will need to invest hours in learning the language. With MySQL, on the other hand, you have to download the software and install it. Thanks to the visual representation, you can easily manage databases using the latest MySQL software.

Creating a database

- A database is a structured set of computerized data with an accessible interface
- The first thing we need to do is to create a database.
- To do so, lets start up our system!
- Type in: **mysql-ctl cli**



Importance of semi-colon

- Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.
- Please add semi-colon after each queries!

First, lets see if there are any existing database!

Type: show databases;

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

Now, let’s create a database call “test_database”

Type: create database test_database;

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> create database test_database;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Now, let’s delete that database!

Type: drop database test_database;

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> create database test_database;
Query OK, 1 row affected (0.00 sec)

mysql> drop database test_database;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Now, let’s create back the database call “test_database”

Type: create database test_database;

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> create database test_database;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Now, we will need to specify which database to use

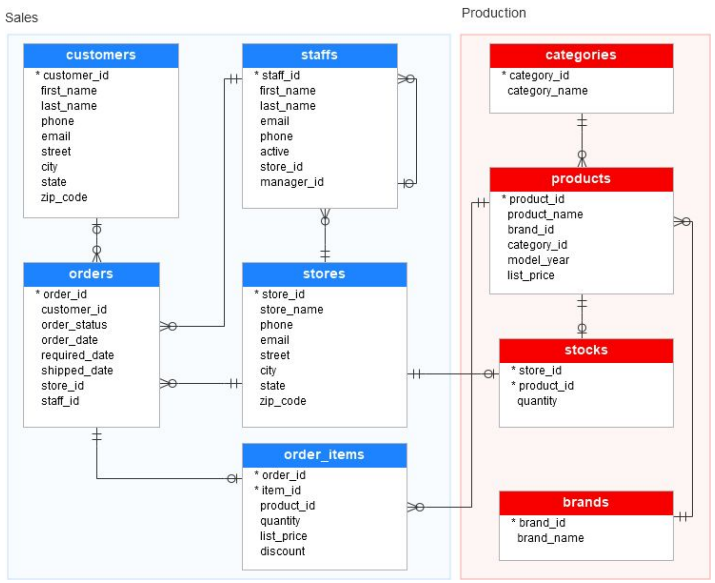
Type: use test_database;

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> use test_database;
Database changed
mysql>
```

CREATE TABLE Statement

- A **table** is a data structure that organizes information into rows and columns. It can be used to both store and display data in a structured format.
- For example, **databases** store data in **tables** so that information can be quickly accessed from specific rows.



CREATE TABLE statement (continue)

- **Syntax for CREATE TABLE**
  - CREATE TABLE <table_name> (column1 datatype, column2 datatype, column3 datatype…);
    - Column1, column2 and column3 represents the names of the columns
    - Datatype represents type of the data the column can hold (varchar, integer, datetime, float, etc). Details on datatype can be found here.

CREATE TABLE exercise

- Create a table call ”orders” with the following columns:
  - OrderID
  - ItemName
  - LastName
  - FirstName
  - Price
  - Date

Create an Orders table

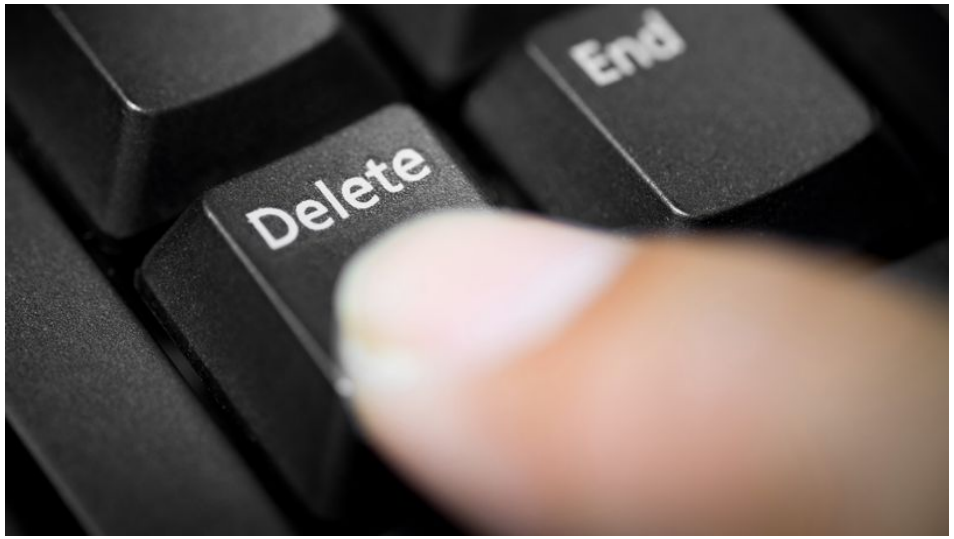
- Type in: **create table orders (OrderID int, ItemName varchar(255), LastName varchar(255), FirstName varchar(255), Price float, Date datetime default now());**
- The **NOW()** function returns the current date and time. We want it to be default (which means if we do not put in entry for date, it will use now().

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> create table orders (OrderID int, ItemName varchar(255), LastName varchar(255), FirstName varchar(255), Price float, Date datetime default now());
```

Dropping a table

- To drop an existing table in a database (eg the order table), type in:
  - Drop table orders;
  - **Do note that SQL is not CASE-SENSITIVE. This means you can type anything in lower character or upper character or mix and it won't make any difference.**
  - **You do not need to drop the order table this time.**



INSERT INTO

- The INSERT INTO statement is used to insert new records in a table.
- This is done by specifying both the column names and the values to be inserted:
  - INSERT INTO *table name* (*column1*, *column2*, *column3*,….) VALUES (*value1*, *value2*, *value3*,…)

Now, try to insert into orders, order key =1, ItemName = ‘pretzel’, LastName = ‘Hudson’, FirstName = ‘Stanley’, price = 3.50, date = ‘2020-03-12 14:25:00’

INSERT INTO exercise

- Type in: **INSERT INTO orders(OrderID, ItemName, LastName, FirstName, price) VALUES (1, 'pretzel', 'Hudson', 'Stanley', 3.50);**
- Use: **SELECT * from orders;** to view entry.
  - * represents all.

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> INSERT INTO orders(OrderID, ItemName, LastName, FirstName, price)  VALUES (1, 'pretzel', 'Hudson', 'Stanley', 3.50);
Query OK, 1 row affected (0.01 sec)

mysql> select * from orders;
+---------+----------+----------+-----------+-------+---------------------+
| OrderID | ItemName | LastName | FirstName | Price | Date                |
+---------+----------+----------+-----------+-------+---------------------+
|       1 | pretzel  | Hudson   | Stanley   |   3.5 | 2020-03-27 04:04:57 |
+---------+----------+----------+-----------+-------+---------------------+
1 row in set (0.00 sec)
```

Altering table

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. Eg. Modifying data types of columns, adding a new column, dropping columns, etc
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

Altering table – Adding columns

- To add a column, use the following syntax:
  - ALTER table orders ADD Email varchar(255);
  - Do note that you do not need to capitalize ALTER or ADD. It will work with all lowercase. I just use it for better readability. Imagine if you have a huge chunk of code, capitalizing helps me to understand how each code function.
  - **Now, I need you to add that email column into goormIDE.**

Altering table – Adding columns

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> create table orders (OrderID int, ItemName varchar(255), LastName varchar(255), FirstName varchar(255), Price float(2,2), Date Datetime);
Query OK, 0 rows affected (0.07 sec)

mysql> ALTER table orders ADD email varchar(255);
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Altering table- Dropping columns

- To drop a column, use the following syntax:
  - ALTER table orders DROP column email;
  - **Now, I need you to drop the column containing email**

Altering table- Dropping columns

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> ALTER table orders DROP column email;
Query OK, 0 rows affected (0.19 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Altering table- Modifying column datatype

- To alter the data type of a column, use the following code:
  - ALTER TABLE orders MODIFY COLUMN date int;
    - Here, I am changing the date column to integer
    - To view datatype of a table use the following code:
      - desc orders;
      - You should be able to see the type for date to be “int”
      - Now, I need you to change back the datatype of date to datetime.

```
mysql> alter table orders modify column Date int;
Query OK, 0 rows affected (0.21 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc orders;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| OrderID   | int(11)      | YES  |     | NULL    |       |
| ItemName  | varchar(255) | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Price     | float(2,2)   | YES  |     | NULL    |       |
| Date      | int(11)      | YES  |     | NULL    |       |
```

Altering table- Modifying column datatype

Type: ALTER TABLE orders MODIFY COLUMN Date datetime;

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> ALTER TABLE orders MODIFY COLUMN Date datetime;
Query OK, 0 rows affected (0.35 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc orders;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| OrderID   | int(11)      | YES  |     | NULL    |       |
| ItemName  | varchar(255) | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Price     | float(2,2)   | YES  |     | NULL    |       |
| Date      | datetime     | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql>
```

SQL Constraints

- Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

SQL Constraints- Examples

- **NOT NULL**: Ensures that a column has no NULL value
- **UNIQUE**: Ensures that all values in that column are different
- **PRIMARY KEY**: A field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.
- **FOREIGN KEY**: A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
- **Default**: Sets a default value for a column when no value is specified

SQL Constraints- NOT NULL

- By default, a column can hold NULL values if nothing is inserted into that column
- Now, let’s create a table call location with column ID, address. These columns will not accept NULL values.

SQL Constraints- NOT NULL

Type: CREATE TABLE address(ID int not null, address varchar(255) not null);

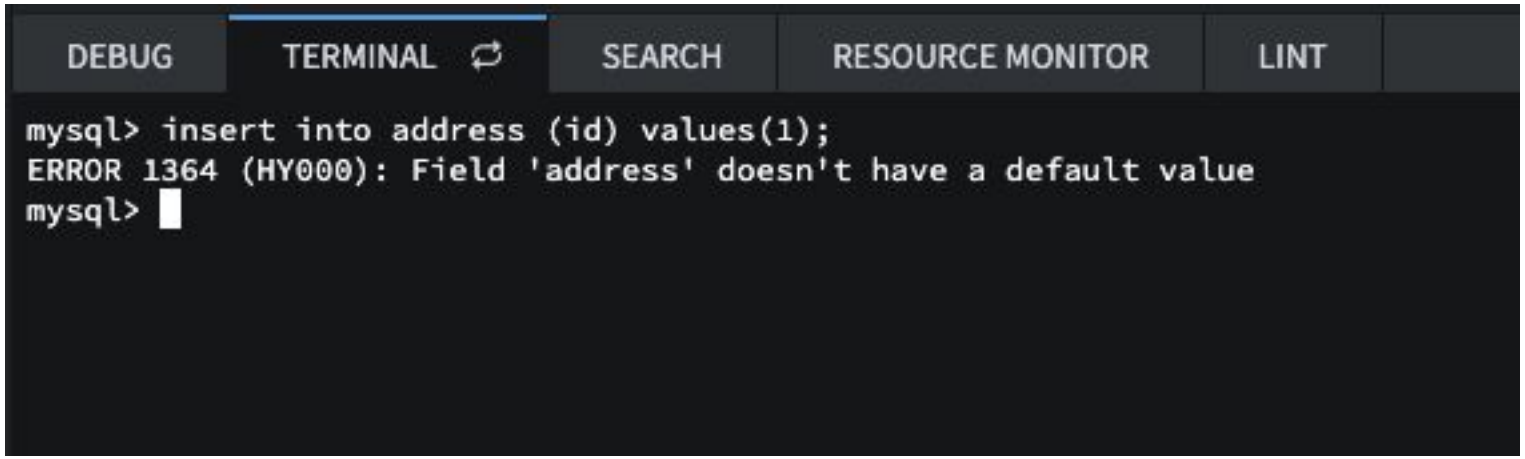
DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> CREATE TABLE address(ID int not null, address varchar(255) not null);
Query OK, 0 rows affected (0.08 sec)

mysql> desc address;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| ID      | int(11)      | NO   |     | NULL    |       |
| address | varchar(255) | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

SQL Constraints- NOT NULL

Type: **insert into address (id) values(1);**

- **You will see an error as you did not add entry for address.**
- **NOT NULL forces user to input in that column**



SQL UNIQUE

- Ensures that all values in a column are different
- Variable like ID has to be unique since each transaction is different.
- Now, let’s alter the table to add in UNIQUE for the ID column in table address.

SQL UNIQUE- Exercise

- Type in: **ALTER TABLE address ADD UNIQUE (ID);**
  - **You will notice that the key field is PRI. That's primary key. I will explain in the next few slides.**

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> ALTER TABLE address ADD UNIQUE (ID);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc address;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| ID      | int(11)      | NO   | PRI | NULL    |       |
| address | varchar(255) | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

SQL Primary Key

- The primary key uniquely identifies each record in a table.
  - It must contain UNIQUE values and no NULL values
  - Each table can only have one primary key

  - The syntax is:
    - CREATE TABLE <table name> (ID int not null, name varchar(255), Age int, Primary key (ID));
    - Now alter the table address and add a primary key to ID

SQL Primary Key- Exercise

Type: ALTER table address add primary key(ID);

```
DEBUG    TERMINAL    SEARCH    RESOURCE MONITOR    LINT

mysql> ALTER table address add PRIMARY KEY(ID);
Query OK, 0 rows affected (0.87 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc address;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| ID      | int(11)      | NO   | PRI | NULL    |       |
| address | varchar(255) | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
2 rows in set (0.04 sec)

mysql>
```

SQL Foreign key

- Foreign key is required to link 2 tables together. If you look into the 2 tables:
- Person table and order table

PersonID	LastName	FirstName	Age
1	Hudson	Stanley	45
2	Scott	Michael	41
3	Schrute	Dwight	39

OrderID	Item	PersonID
1	Beets	3
2	Pretzel	1
3	Magic kit	2

- The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.
- The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.
- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

SQL Foreign key

- Syntax (with reference to previous slide)
  - CREATE TABLE orders (OrderID int NOT NULL PRIMARY KEY, Item varchar(255), PersonID int, FOREIGN KEY (PersonID) REFERENCES Person(PersonID));
  - Now let’s get our hands dirty with some exercise!
    - I need you to drop 2 tables: orders and address
    - Create a table call person with the following columns: PersonID (INT NOT NULL PRIMARY KEY), LastName (VARCHAR(255)), FirstName (VARCHAR(255)) and Age (INT).
    - Create another table call orders with the following columns: OrderID (INT NOT NULL PRIMARY KEY), Item (VARCHAR(255)), PersonID (FOREIGN KEY, referencing person(PersonID))

SQL Foreign Key

```
DEBUG   TERMINAL   SEARCH   RESOURCE MONITOR   LINT

mysql> DROP TABLE address, orders;
Query OK, 0 rows affected (0.21 sec)
```

```
DEBUG   TERMINAL   SEARCH   RESOURCE MONITOR   LINT

mysql> CREATE table person(
    -> PersonID int auto_increment primary key,
    -> first_name varchar(100),
    -> last_name varchar(100),
    -> age int
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql>
mysql> CREATE table orders(
    -> OrderID int auto_increment primary key,
    -> item varchar(255),
    -> Person_ID int,
    -> foreign key(Person_ID) references person(PersonID));
Query OK, 0 rows affected (0.08 sec)

mysql>
```

```
DROP TABLE address, orders;

CREATE table person(
PersonID int auto_increment primary key,
first_name varchar(100),
last_name varchar(100),
age int
);

CREATE table orders(
OrderID int auto_increment primary key,
item varchar(255),
Person_ID int,
foreign key(Person_ID) references person(PersonID));
```

SQL Foreign Key

- You may notice that I use auto_increment. That will help me to number PersonID from 1 onwards automatically.
- Now let's insert similar values in there from these 2 tables.

PersonID	LastName	FirstName	Age
1	Hudson	Stanley	45
2	Scott	Michael	41
3	Schrute	Dwight	39

OrderID	Item	PersonID
1	Beets	3
2	Pretzel	1
3	Magic kit	2

SQL Foreign Key

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> INSERT INTO person(first_name, last_name, age) VALUES ('Stanley', 'Hudson', 45), ('Michael', 'Scott', 41), ('Dwight', 'Schrute', 39);
Query OK, 3 rows affected (0.08 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO orders(item, Person_ID) VALUES ('Pretzel',1), ('Magic kit', 2), ('Beet', 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from person;
+----------+------------+-----------+------+
| PersonID | first_name | last_name | age  |
+----------+------------+-----------+------+
|        1 | Stanley    | Hudson    |   45 |
|        2 | Michael    | Scott     |   41 |
|        3 | Dwight     | Schrute   |   39 |
+----------+------------+-----------+------+
3 rows in set (0.00 sec)

mysql> select * from orders;
+---------+----------+----------+
| OrderID | item     | Person_ID |
+---------+----------+----------+
|       1 | Pretzel  |        1 |
|       2 | Magic kit |        2 |
|       3 | Beet     |        3 |
+---------+----------+----------+
3 rows in set (0.01 sec)
```

SQL Default

- The DEFAULT constraint is used to provide a default value for a column.
- The default value will be added to all new records IF no other value is specified.

- Syntax
  - CREATE TABLE <table name> (ID INT NOT NULL, item varchar(255) DEFAULT ‘chicken’);
    - Hence if I do not input anything in item, the default option will be chicken.
    - Let’s alter orders and set default as ‘chicken’ for item

SQL Default

DEBUG | TERMINAL | SEARCH | RESOURCE MONITOR | LINT

```
mysql> ALTER TABLE orders ALTER item set default 'chicken';
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Part 2 will be out soon!