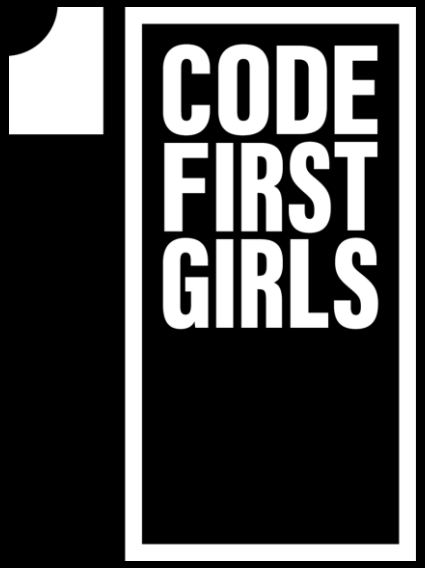
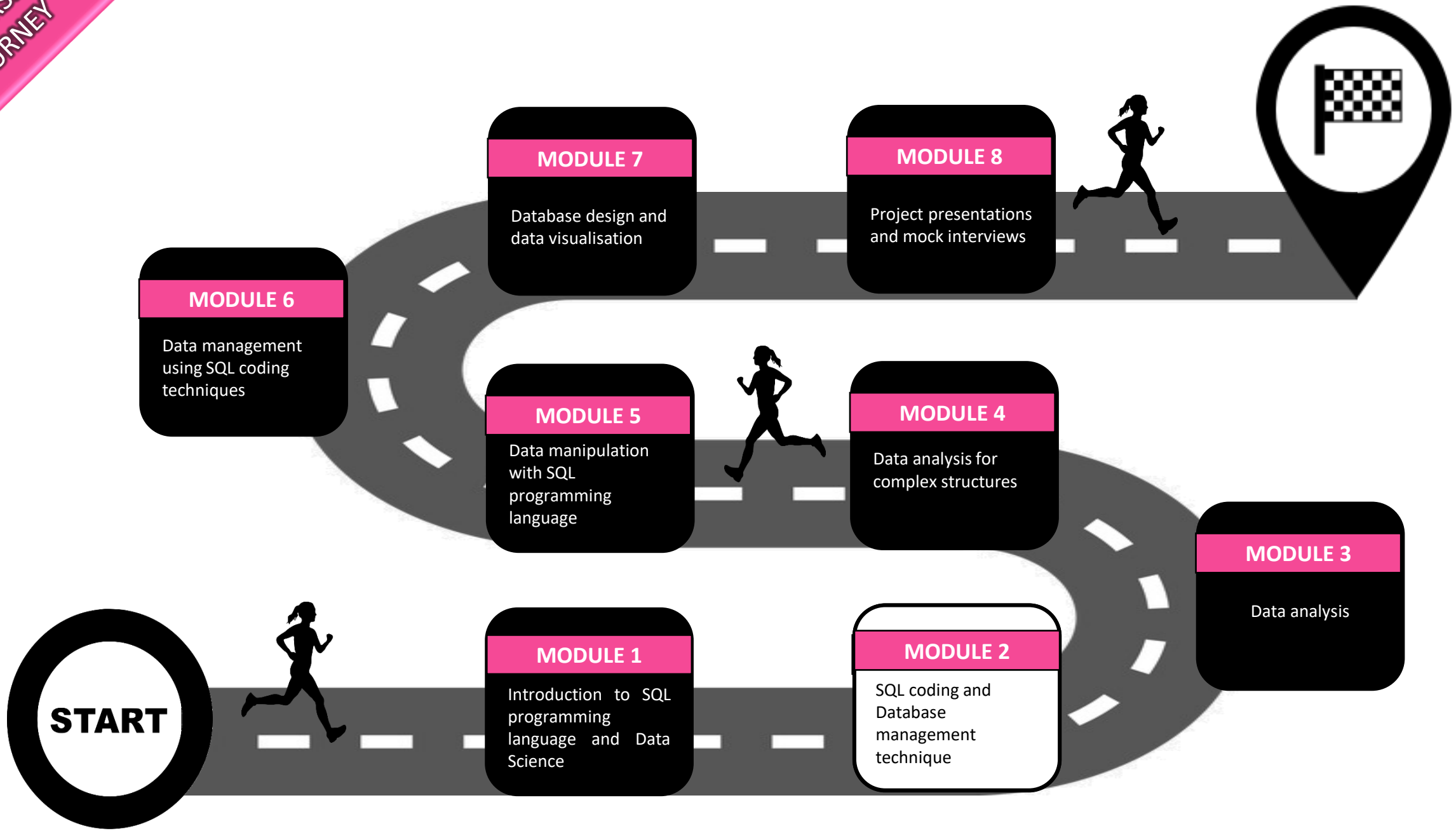
WELCOME TO CFG

YOUR INTRODUCTION TO DATABASES & SQL PROGRAMMING LANGUAGE



TECH OPENS UP LIMITLESS OPPORTUNITIES FOR GIRLS

COURSE JOURNEY



02

SQL CODING
DB MANAGEMENT TECHNIQUE

1. DB design and management:
   - **Constraints on a table**
   - **Primary and Foreign keys**
   - **Normalisation**.

2. SQL Coding:
   - **Data Modification Techniques**
   - **Data Retrieval Techniques** (SELECT statements)

SELECT DISTINCT

- There are many ways that enable us to constrain the number of results returned by our query
- DISTINCT Qualifier
- It is the keyword, which means that in our query we are asking for a unique set of results
- In other words we want non-repeating values in the result columns to be returned

SYNTAX

Table name Alias

SELECT DISTINCT
<alias>.<column_name>,

FROM <table_name>
<alias>;

EXAMPLE

What are the fist names of people in my class?

first_name
Julie
Mary
Mary
Joanna
Julie

first_name
Julie
Mary
Joanna

```
SELECT
p.first_name,
FROM person p;
```

```
SELECT DISTINCT
p.first_name,
FROM person p;
```

WHERE

- The WHERE clause is a constraint that can be applied to the result set
- The WHERE clause describes the **conditions** to match for rows to qualify for result set
- It comes after the FROM statement
- It contains **Boolean** expressions
- Only rows that match a condition are selected for the result set

SYNTAX

```
SELECT
<alias>.<column_name>,

FROM <table_name>  <alias>;
```

EXAMPLE

What is the surname of all my classmates who are called Mary?

```
SELECT                          ← SELECT clause
p.surname
FROM person p                   ← FROM clause
WHERE p.name = 'Mary'           ← WHERE clause
```

PRACTICE

Use the database PARTS that we created and populated at home

WRITE THE FOLLOWING QUERIES

- Using the table ‘parts’, return all unique part names. What happens if we want to return all unique parts and their id number? Why?

- Refer to the table ‘projects’ and return all projects that are run in London.

DATABASE NORMALISATION

- The idea behind normalisation is to organise a database into tables in such way that a table is created about one specific topic only.

- The main reasons to normalise a database are:
  - to minimise duplicate data,
  - to minimize or avoid data modification issues
  - to simplify queries

- There are three common forms of database normalization:
  - 1st, 2nd, and 3rd normal form or
  - 1NF, 2NF, and 3NF respectively

NB please read about 3 forms of normalisation:
https://www.complexsql.com/database-normalization/

NORMALISATION EXAMPLE 1

- We want to design our DB that it can answer as many ‘questions’ as possible including the ones we may ask in the future. Design it in a way that we can always add more data to it and easily modify existing data if necessary.

Table 1 – NO normalisation applied

Employee No	Employee Name	Department
1	Mary	FINANCE,TAX
2	Edith	HR
3	Anna	ADMIN

Table 2.1 – normalised

Employee No	Employee Name
1	Mary
2	Edith
3	Anna

Table 2.2 – normalised

Employee No	Department
1	FINANCE
1	TAX
2	HR
3	ADMIN

NORMALISATION EXAMPLE 2

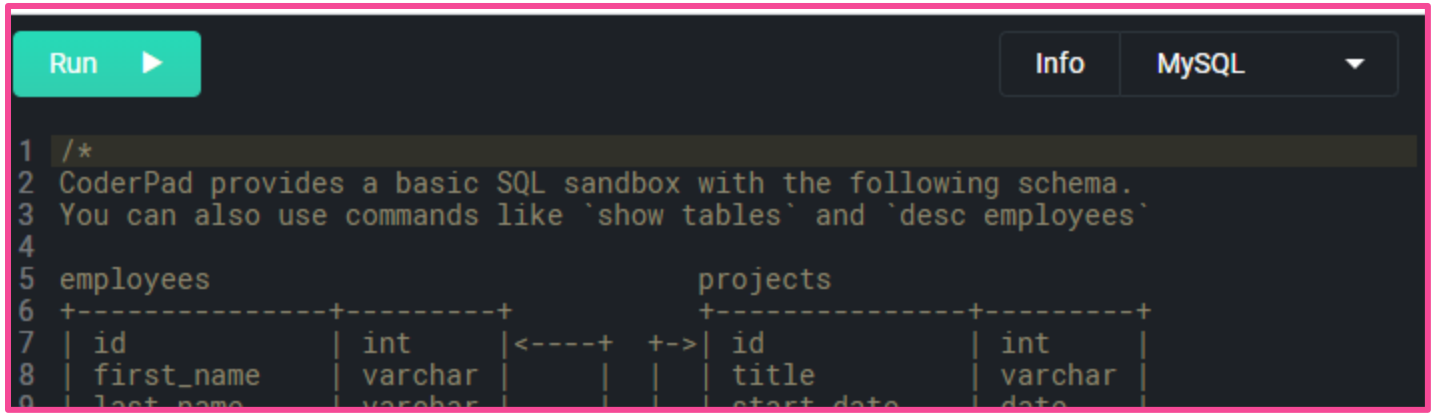
Customer Name	Customer Address	Customer Tel No.	Product Name	Unit Cost	Quantity	Total Cost
Alex Wilson	1318 Scenic Avenue, Bothel	697-555-0142	Men's Sports Shorts, S	15.5	2	31
Alex Wilson	1318 Scenic Avenue, Bothel	697-555-0142	Water Bottle - 30 oz	1.5	3	4.5
Alex Wilson	1318 Scenic Avenue, Bothel	697-555-0142	LL Mountain Handlebars	19.76	2	39.52
Emily Brown	628 Muir Road, Los Angeles	708-555-0141	Long-Sleeve Logo Jersey, S	38.49	1	38.49
Emily Brown	628 Muir Road, Los Angeles	708-555-0141	Sport-100 Helmet, Black	13.08	2	26.16
Emily Brown	628 Muir Road, Los Angeles	708-555-0141	LL Mountain Handlebars	19.76	3	59.28

source: http://salinaitmind.blogspot.com/2012/10/data-warehouse-data-normalization.html

PRACTICE

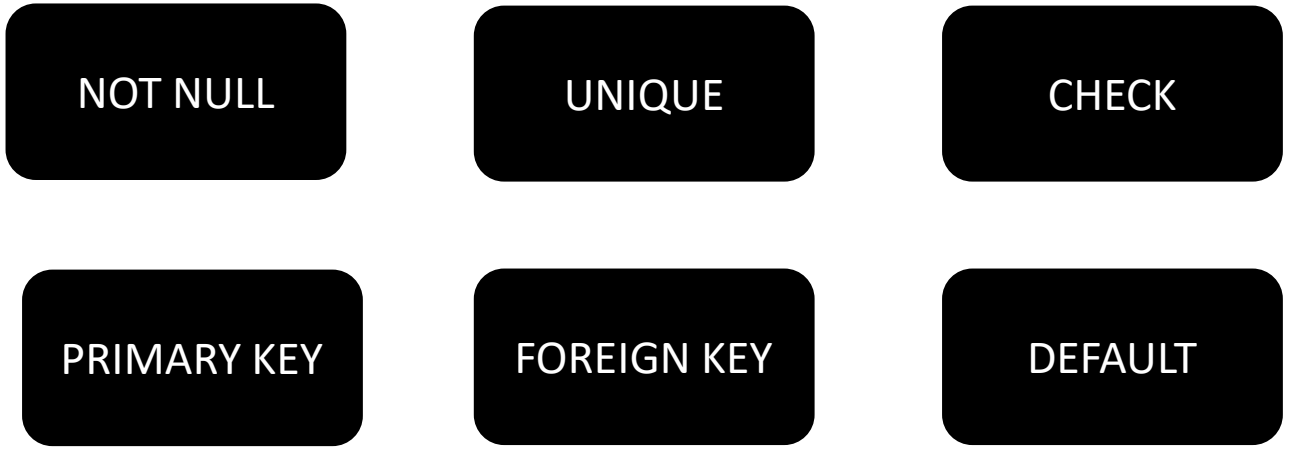
LET'S REVIEW THE DB DESIGN IN OUR SANDBOX

https://coderpad.io/sandbox



SQL CONSTRAINT TYPES

- Constraints are the rules that we can apply on the type of data in a table.
- In other words, we can specify the limit on the type of data that can be stored in a particular column in a table. Using constraints ensures the accuracy and reliability of the data in the table.
- If there is a mismatch or any violation between the constraint we set and the data, then the action that we are trying to perform would be aborted.



NB please read about constraints: https://www.studytonight.com/dbms/sql-constraints.php

PRIMARY KEY

- primary key is a single field or combination of fields that **uniquely defines a record**

ONLY ONE PRIMARY KEY IN A TABLE

- Must be NOT NULL
- Can be a multiple columns (compound key)
- Can be defined in either a CREATE TABLE statement or an ALTER TABLE statement

NULL	NOT NULL
Default for a column definition	Must be specified on column definitions
It means that we are allowed to insert NULL values	It means we are not allowed to insert NULL values. Inserting a NULL would raise an error!

SYNTAX

```
CREATE TABLE <table_name>
(col1 Type KEY DEFINITION,
 col2 Type,
 col3 Type);
```

EXAMPLE

```
CREATE TABLE customers
(customer_id INTEGER PRIMARY KEY,
 name VARCHAR(50),
 surname VARCHAR(50) NOT NULL,
 telephone INTEGER);
```

← CREATE TABLE + TABLE NAME

← PRIMARY KEY

← COLUMNS

SYNTAX

```
CREATE TABLE <table_name>
(col1 Type,
 col2 Type,
 col3 Type,
 CONTRAINT
 <constraint_name>
 <constraint_type>
(<col_that_it_applies_to>)
);
```

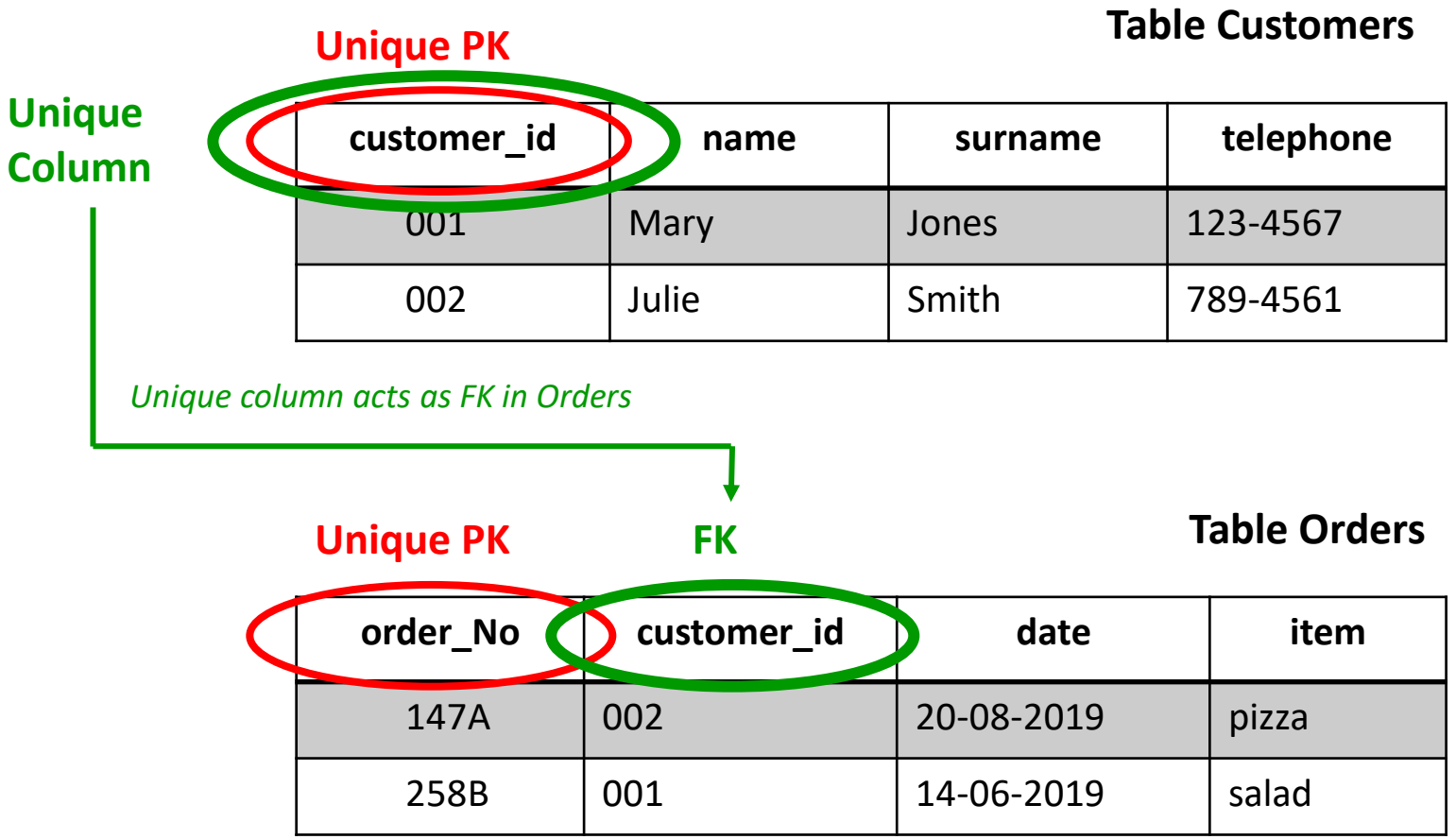
EXAMPLE

```
CREATE TABLE customers
(customer_id INTEGER,
 name VARCHAR(50),
 surname VARCHAR(50) NOT NULL,
 telephone INTEGER,
CONSTRAINT
pk_ customer _id
PRIMARY KEY
(customer _id)
);
```

← CREATE TABLE + TABLE NAME

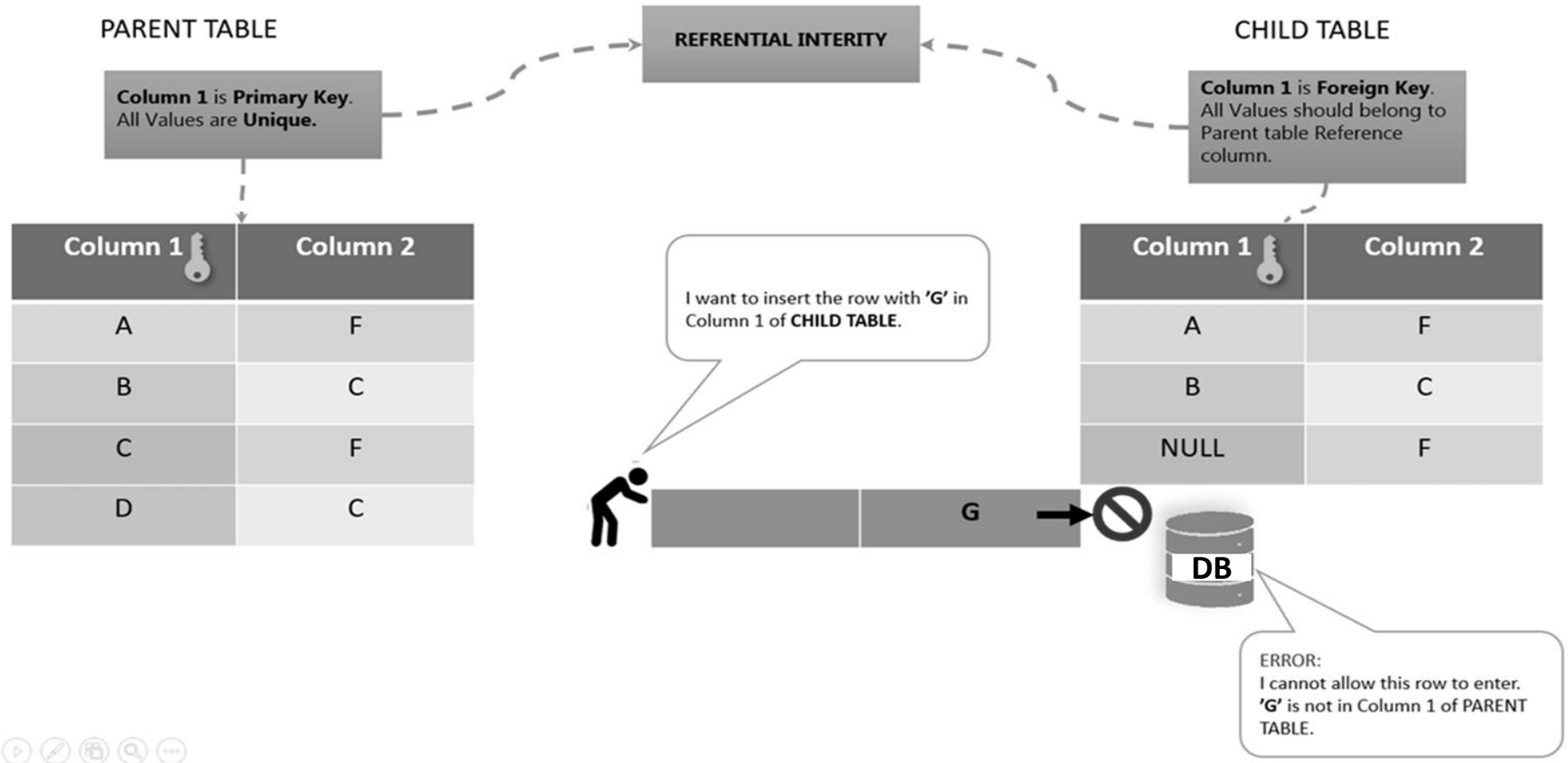
← CONSTRAINT

FOREIGN KEY



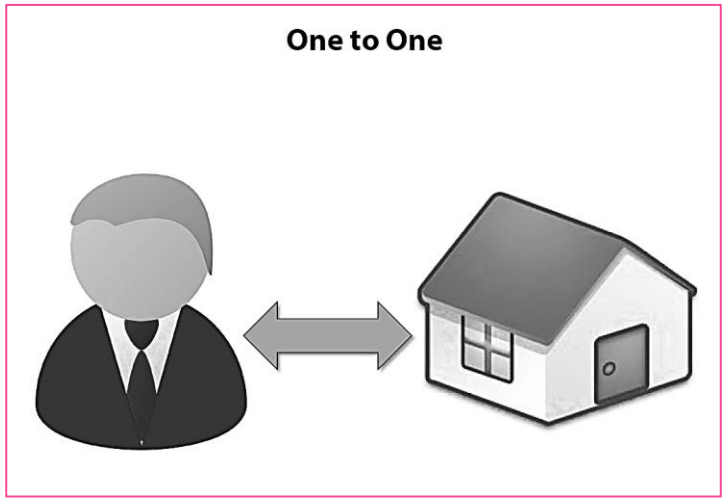
NB please read about Foreign Key: http://www.mysqltutorial.org/mysql-foreign-key/

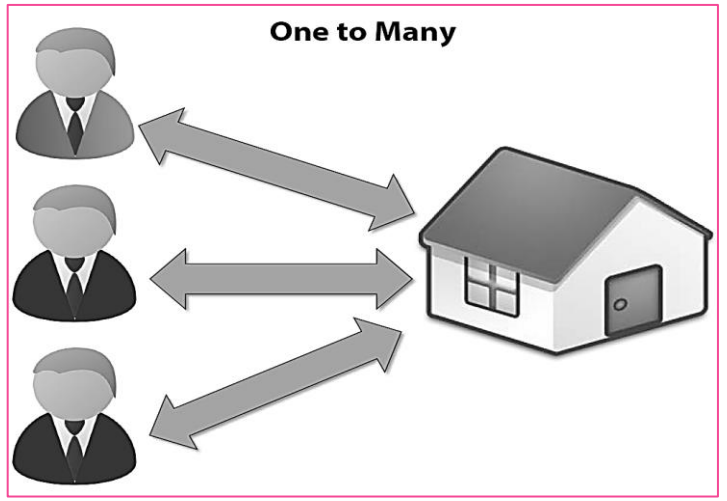
DATA INTEGRITY

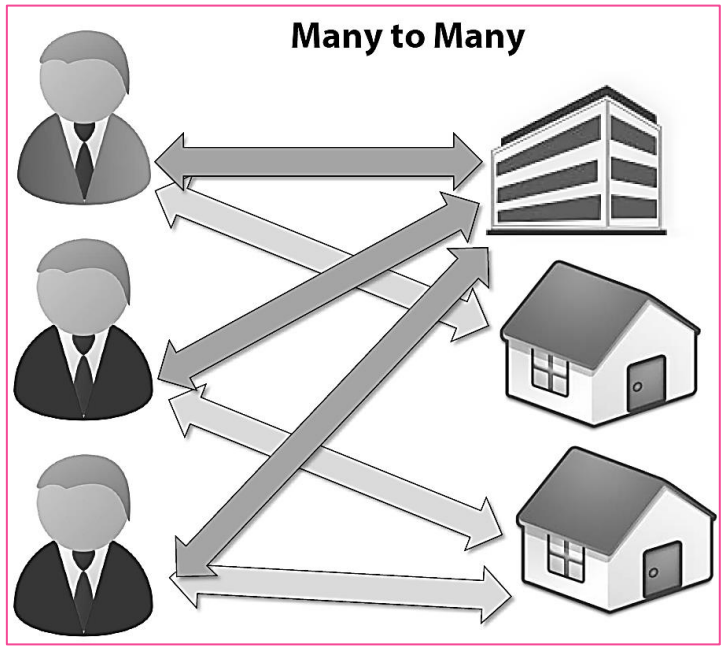


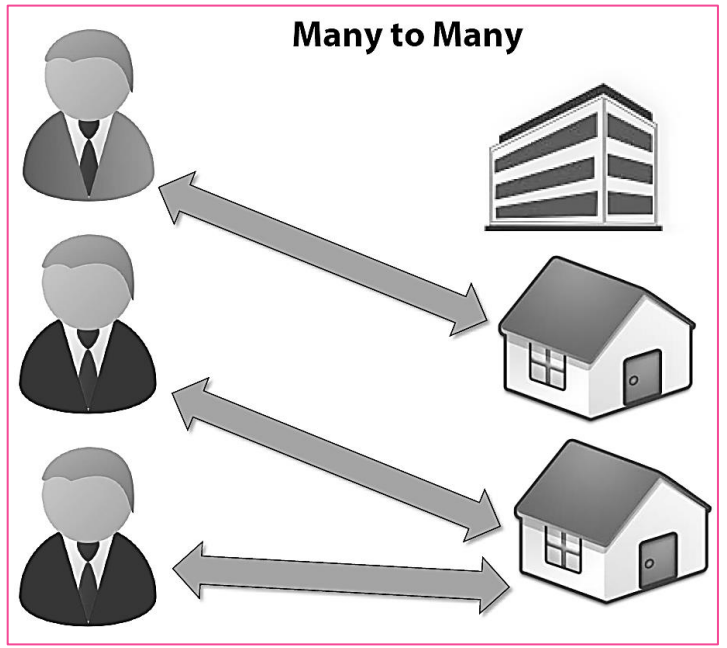
source: guru99.com

DATABASE RELATIONSHIPS









PRACTICE

Today we are going to be pizza makers, bakers and small restaurant owners! We accept orders online or by telephone and deliver pizza to our customers.

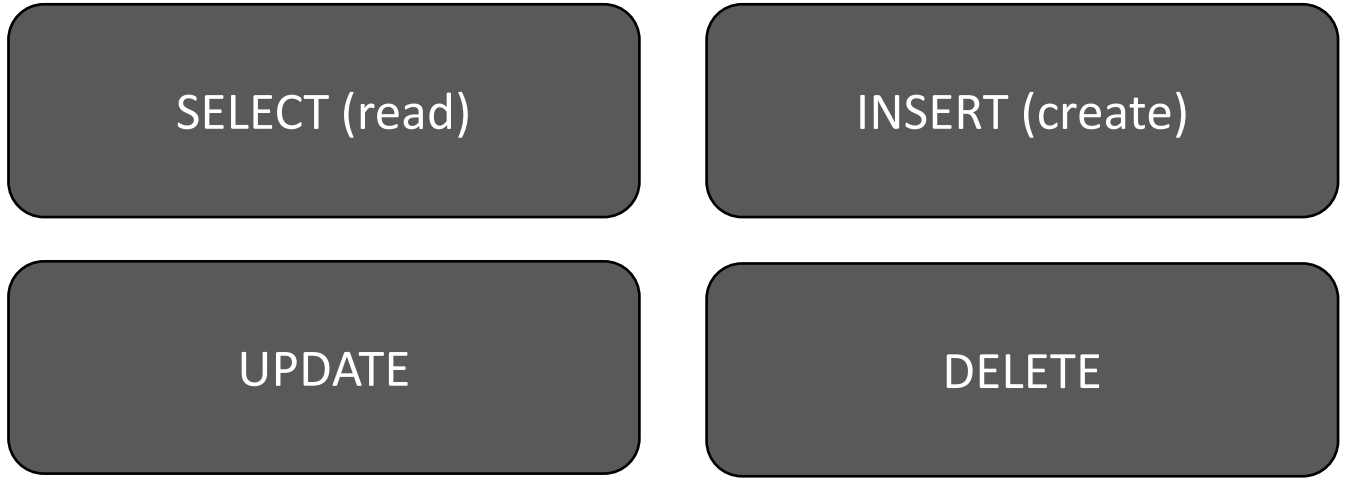
We need to create a database to hold information about our customers, so we can keep records of their names, addresses , phone numbers, email addresses and any other useful information like placed orders.

TASKS

- Design and create a relational normalised database called **customers**.
- Set reasonable **primary keys** to the tables.
- Set NOT NULL **constraints** on the columns that you think must have values.
- Let’s do it together!

CORE COMMANDS DDL

- **DDL** stands for "Data Definition Language“.
- It is a subset of **SQL statements** that change the structure of the database schema.
- Typically structural changes of the database schema refer to creating, deleting, or modifying schema objects such as databases or tables.



NB please read more about DDL: https://www.w3schools.in/mysql/ddl-dml-dcl/

UPDATE

- Modifies column(s) in a single table
- WHERE clause dictates which rows
- SET keyword follows table name

SYNTAX

```
UPDATE table_name
SET
table_name.col1 = new_value
WHERE
table_name.col2 = value;
```

EXAMPLE

```
UPDATE contacts
SET
contacts.mobile = 123456789
WHERE
contacts.surname = ‘Andrews’
```

← UPDATE COMMAND

← TABLE NAME

← SET KEYWORD

← VALUES

← WHERE CLAUSE

DELETE

- DELETES one or more rows in a table
- Permanent!
- DELETE FROM is actual full command
- WHERE clause is critical!

SYNTAX

```
DELETE FROM table_name;
```

```
DELETE FROM table_name
WHERE
table_name.col = value;
```

EXAMPLE

- DELETE FROM customers;

← DELETE COMMAND

BAD PRACTICE ☹

- DELETE FROM customers
- WHERE
- customers.id = 007;

← DELETE COMMAND

← WHERE CLAUSE

GOOD PRACTICE ☺

ALTER TABLE

- Used to change an existing table
- Add/remove column
- Change column data type
- Change column constraints
- Must comport with current data

SYNTAX

```
ALTER TABLE <table_name>
ADD CONTRAINT
 <constraint_name>
 <constraint_type>
(<col_that_it_applies_to>)
REFERENCES
<table_name2>
(<col2_that_it_applies_to>)
);
```

EXAMPLE

```
ALTER TABLE orders
ADD CONSTRAINT
fk_customer_id
FOREIGN KEY
(customer_id)
REFERENCES
customers
(customer_id);
```

← CREATE TABLE + TABLE NAME

← CONSTRAINT

DROP TABLE

- Removes table and all data from database
- BE CAREFUL!
- Error if table is a foreign key to another table

SYNTAX

DROP TABLE <table_name>;

EXAMPLE

DROP TABLE customers;

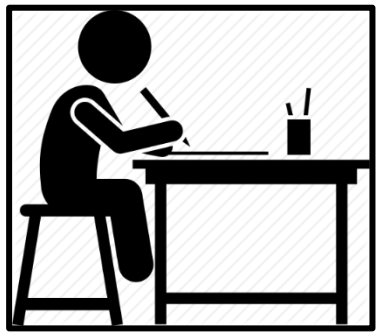
PRACTICE

We have our pizzeria customers database. Let’s modify some tables in the database, so we add Foreign Keys to tables and define relationships between our tables.

TASKS

- Add some data to the tables in the **customers database**
- Alter tables **email_address** and **phone_number** in the **customers database** by adding **Foreign keys** that reference **Primary keys** from relevant tables.
- Remove the table called **orders** from our database.

HOMEWORK



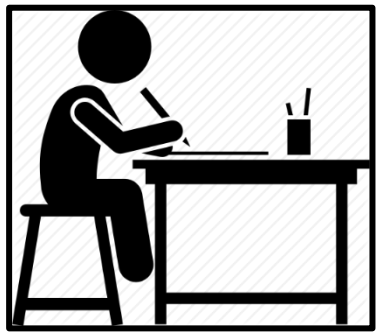
- Revise the slides to re-cap all materials from Module2.
- Read about **Normalisation**: https://www.complexsql.com/database-normalization/
- Read about **DDL**: https://www.w3schools.in/mysql/ddl-dml-dcl/
- Read about **Foreign Key**: http://www.mysqltutorial.org/mysql-foreign-key/
- Read about Constraints: https://www.studytonight.com/dbms/sql-constraints.php

USE PARTS DB TO WRITE THE FOLLOWING QUERIES

- Find the name and weight of each red part
- Find all unique supplier(s) name from London.

(*you must submit the code and results)*

HOMEWORK

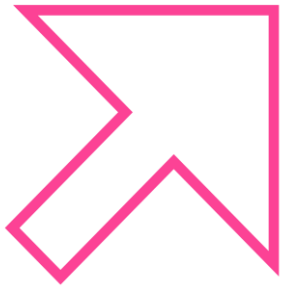


- Create a new database called **SHOP –** we will be using it during next lesson.
- Add a new table called **SALES1**. It should look like this:

Store	Week	Day	SalesPerson	SalesAmount	Month
London	2	Monday	Frank	56.25	May
London	5	Tuesday	Frank	74.32	Sep
London	5	Monday	Bill	98.42	Sep
London	5	Saturday	Bill	73.90	Dec
London	1	Tuesday	Josie	44.27	Sep
Dusseldorf	4	Monday	Manfred	77.00	Jul
Dusseldorf	3	Tuesday	Inga	9.99	Jun
Dusseldorf	4	Wednesday	Manfred	86.81	Jul
London	6	Friday	Josie	74.02	Oct
Dusseldorf	1	Saturday	Manfred	43.11	Apr

THANK YOU
HAVE A GREAT WEEK!

CODE FIRST GIRLS



REFERENCE MATERIALS

REVISION

FOREIGN KEY

- A foreign key is a field in a table that matches another field of another table. A foreign key places constraints on data in the related tables.
- A foreign key can be a column or a set of columns. The columns in the child table often refer to the primary key columns in the parent table.
- A table may have more than one foreign key, and each foreign key in the child table may refer to a different parent table.

NB please read about Foreign Key: http://www.mysqltutorial.org/mysql-foreign-key/

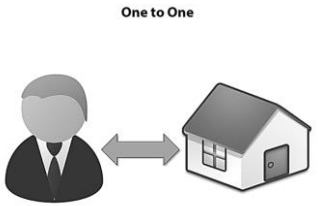
DATABASE RELATIONSHIPS

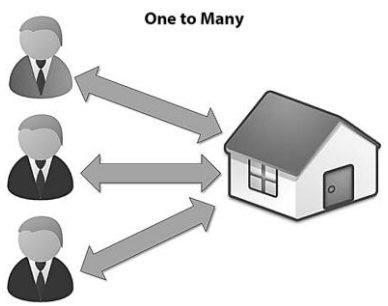
When creating a database, common sense dictates that we use separate tables for different types of entities.

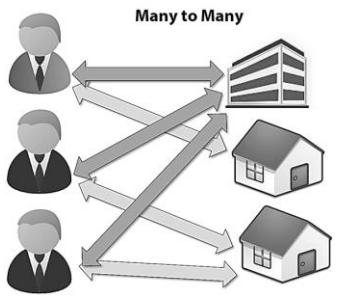
Some examples are: customers, orders, items and so on. But we also need to have relationships between these tables. For instance, customers make orders, and orders contain items.

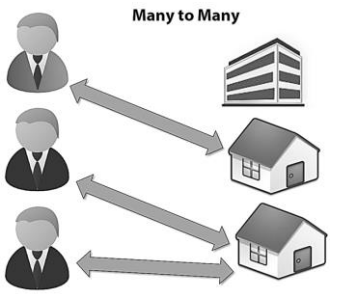
There are several types of database relationships:

- One to One Relationships
- One to Many
- Many to One Relationships
- Many to Many Relationships









QUICK SUMMARY

PRIMARY KEY

- Unique identifier of row
- One per table
- Does not allow NULL
- Single or multiple columns (composite columns)

FOREIGN KEY

- Columns in a table that refer to a Primary Key of another table
- Enforces referential integrity
- Foreign key reinforces relationships between tables:
  - One-to-one
  - One-to-many
  - Many-to-many