Exercise 5 – Create tables with Indexes and Foreign Keys Code breakdown

* This script contains three distinct blocks of code to create the three tables. Each block is terminated by a semicolon.
* Take note of the order the tables are created in the script
  + **eventvenues** is created first, the table its Foreign Key refers to, **states**, is already in the database however, the **clubevents** table has references to **eventvenues** so therefore **eventvenues** MUST be created first as it needs to exist before **clubevents** can be created.
  + The **eventtype** has no Foreign Keys referencing it so it can be created at any point, we have put it last here.

The target or parent table in a Foreign Key relationship must exist before the Foreign Key can be created

* New commands in the **eventvenues** block are also present in the other two blocks so we will only breakdown the code in **eventvenues** for lines 17 to 24 (Figure 3.16), starting at the line beginning with **INDEX**

INDEX `FK\_EventVenue\_States\_idx` (`VenueState` ASC),

* This command instructs the server to create a standard Index with the name that is in the back ticks. The brackets contains the field the index is to be created on and the ASC indicates the sort order for the index is ascending, the field can be of any datatype and the sort order will sort appropriately for the datatype. Note, the **clubevents** table has an index on the **EventDate** field that is set to sort in descending order, as it is a Date datatype, sorting in descending order will cause the most recent dates to be at the top so we can access them faster. Also note, this index is for the Foreign Key named **FK\_EventVenue\_States**, you can leave this out and the system will create it automatically when the Foreign Key is created. However, if the index was NOT on a Foreign Key field then you will need to include it, it will not be generated by the system.

UNIQUE INDEX `Idx\_VenueName` (`VenueName` ASC),

* This is a UNIQUE index, the same rules regarding the structure of the command apply as for the previous standard index however the values in the field must be unique, and the same venue name cannot appear twice.

CONSTRAINT `FK\_EventVenue\_States`

FOREIGN KEY (`VenueState`)

REFERENCES `autoclub`.`states` (`ID`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

* This is a actually single command defining the Constraint and Foreign Key.
  + **CONSTRAINT**, initiates the Foreign Key creation, the name of the Foreign Key is in back ticks
  + **FOREIGN KEY**, indicates the field the Foreign Key will be applied to in this table (Child)
  + **REFERENCES**, indicates the database and table the Foreign Key will reference, the actual field referenced is in the brackets. The database, table and field names must be enclosed in back ticks.
* **ON DELETE** and **ON UPDATE**, both define the constraint type to use respectively, in this table we selected **RESTRICT** for both to enforce the rule that the value in the Child tables Foreign Key field MUST be in the Parent tables referenced field otherwise any create or update query will fail, this is to ensure data integrity.

ENGINE = InnoDB;

* This command tells the server what database engine to use for the table. You only need to include this if you wish to use a different engine than database default InnoDB engine. Common options for this are **InnoDB** and **MyISAM**, there are several more, usually you will leave this out and use the default unless you have specific requires to change it. InnoDB is the mySQL default and the recommended engine. The MyISAM engine will lock the entire table when written to, this can severely impact performance for tables with heavy read/write requirements so they are mainly used for read only tables, recommendation, stick with the InnoDB engine and save yourself some hassle and do some research on the engine types MySQL can use.