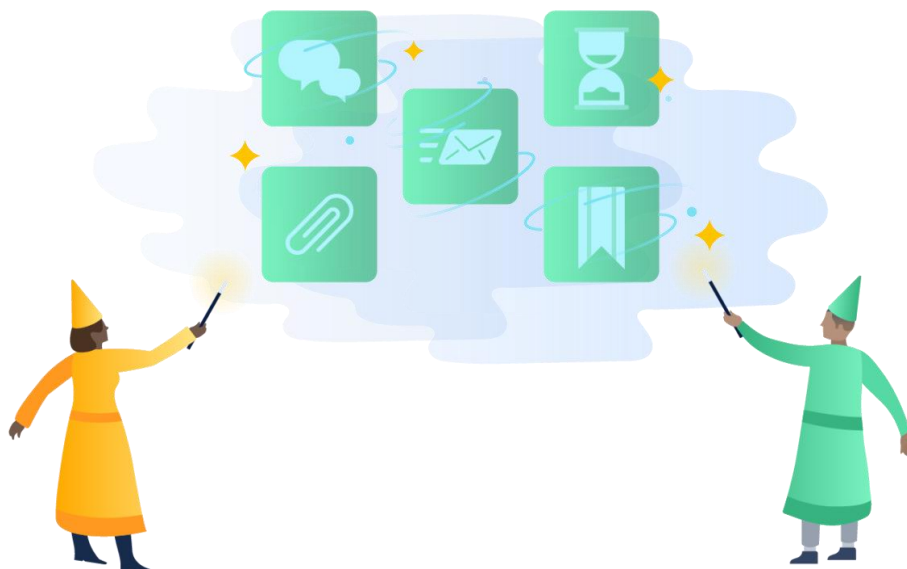




Structured Data Modelling Mini Assignment

Unpacking the *magic* of Data Modelling




Introduction

Welcome to your first mini Data Modelling Assignment for the Tracked Analytics Career Track introducing you to the  dbdiagram.io platform.

As you've gone through the data modelling materials, we've covered a series of data modelling terms. We've also introduced some core concepts known as One to One, One to Many and Many to Many Relationships that you'll be working with in the dbdiagram.io platform.

We've listed some of the common terms below for your reference as well as a high-level guide which explains how you will utilise the dbdiagram.io platform to **programmatically** create your first data model (Entity Relationship Diagram).

Let's take a refresher on the  dbdiagram.io platform terminologies before we dive into creating a practice data model.

Data Modelling Refresher

Creating a table

To create a table in dbdiagram.io, you'll have to make use of SQL which will allow us to create the visual representations of the table. Now you won't learn SQL until the ETL Track Stop, so we've provided the basic syntax and tips you'll need to create your own tables below. The very basic syntax we've provided here is all you'll need to complete the exercises.

Creating a table can be done with the below syntax:

```
Table tableName {  
  Datafield datatype [pk]  
  Datafield datatype  
}
```

Let's break this down now and understand exactly what is happening here.

To instantiate (create) a table, we use the **Table** notation as below:

```
Table tableName {  
}
```

At this point, we have created an **empty table** called tableName. The open braces `{}` indicate the **start** and **end** of our table respectively. Everything included within these braces are the data fields that belong in the table.

Data fields

Once we've instantiated a table. We now need to **fill the table**. We fill a table with **data fields**. **Data fields** are the **columns** that make up the table. Here, we specify what the **data field name is** and what the **data type is**. **Please note that every table should have ONE primary key**.

This is generally the **first data field** in the table and we reference this with the following syntax:

Column_name_of_data_field datatype [pk]

Column_name_of_data_field – This is the name of the data field (i.e. Customer_ID)

Data type – This indicates what data type the data field should belong towards. For example, is the data type an integer, a float, a string variable etc. You can see the full range of data types available in the data type references [here](#).

[pk] – This represents whether the field is the primary key for the table or not.

We can see this all combined in the image below:

```
4 // Creating tables
5 Table CustomerTable {
6     customer_id int [pk] // auto-increment
7
8 }
```

We've created an **example** CustomerTable for you with the below syntax that you can simply copy and write into the code terminal in the dbdiagram.io platform:

```
//// -- LEVEL 1
//// -- Create all the table as per the Data Reference Sheet

// Creating tables
Table CustomerTable {
    customer_id int [pk] // auto-increment
    full_name varchar
    last_name varchar
    frequent_flyer_number varchar
    country_code int
}
```

CustomerTable	
customer_id	int
full_name	varchar
last_name	varchar
frequent_flyer_number	varchar
country_code	int

Creating relationships between different table entities

Creating tables is the first step to building out a data model. However, we also need to be able to connect these tables to each other and understand how these tables are related to each other.

In the material, we covered the concepts of:

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

Let's review this one more time for clarity before looking at how to create this in dbdiagram.io.

One-to-one

The simplest kind of relationship is a one-to-one relationship. Suppose you have a list of people's names, and a list of Tax File Numbers. Each person has only one Tax File Number (TFN_ID), and each Tax File Number is linked to one person (TFN_ID).



To show a One-to-Many relationship in the dbdiagram.io platform, use the below syntax:

Ref: Person.TFN_ID – TaxFileNumber.TFN_ID

One-to-many

A more complex type of relationship is one-to-many/many-to-one. For example, one person may have many bank accounts. However, these many bank accounts are linked to the one primary owner. Another example might be to have a list of works of art and a list of museums, each work of art can only be in one museum at a time, but each museum can have many works of art. Splitting these two lists of entities (museums and works of art) into two tables allows you to store information relevant to each entity. For works of art, this could include information like the artist and date of completion, and for museums, this could include information about the museum's opening hours and its address.



To show a One-to-Many relationship in the dbdiagram.io platform, use the below syntax:

Ref: Person.BankID < BankTable.BankID

Many-to-many

Lastly, entities can also have a many-to-many relationship. Let's say you have a list of books, and a list of authors. Each book may have one or more authors, and each author may have written multiple books. In this case, you have many books related to many authors. Hence the relationship is Many-to-Many.

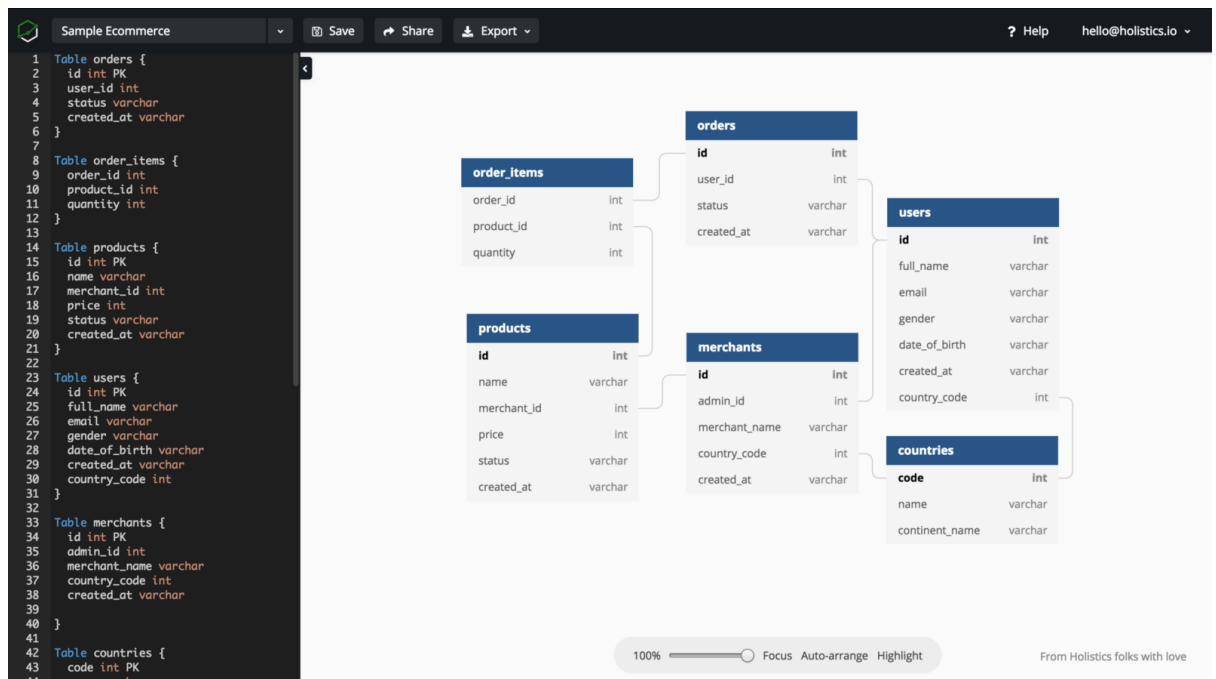


As a data analyst, you will have to familiarise yourself with the different types of data relationships as it will help you to better understand the data you are dealing with via an data model. Remember, data models are simply visual representations of the data that enable us to understand which entities/tables are related to each other which will form the basis of how we create queries where we will analyse our respective data.

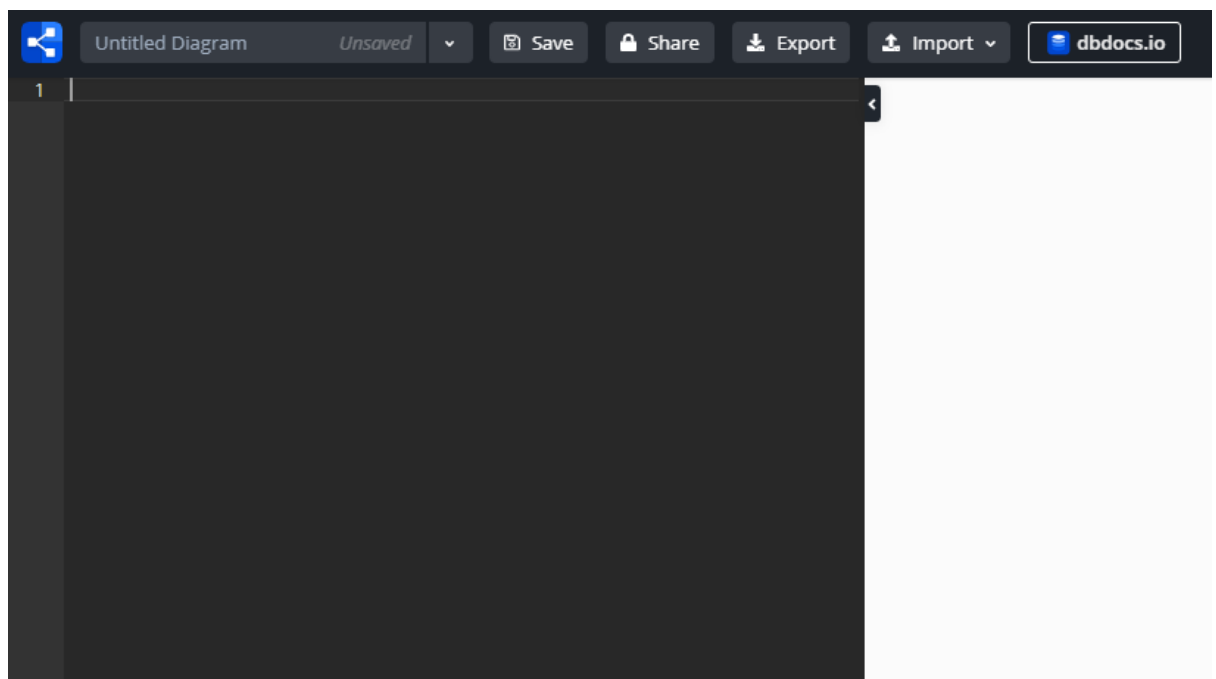
Note: You won't need to create Many-to-Many relationships for this exercise, but for the syntax, refer to the following reference [here](#).

Instructions

1. Using the dbdiagram.io platform and the **eCommerce_Table_Reference document**, create the eCommerce data model. (You'll need to think carefully regarding the **relationships** that link each table together.)



Note: When opening up the dbdiagram.io platform, you will need to delete all the code that is preloaded in the terminal window until the terminal is blank before creating any tables.



2. Save your data model and send a copy of this to your Guide for review.

Great job! You've successfully built your first mini data model conceptual data model using the dbdiagram.io platform!