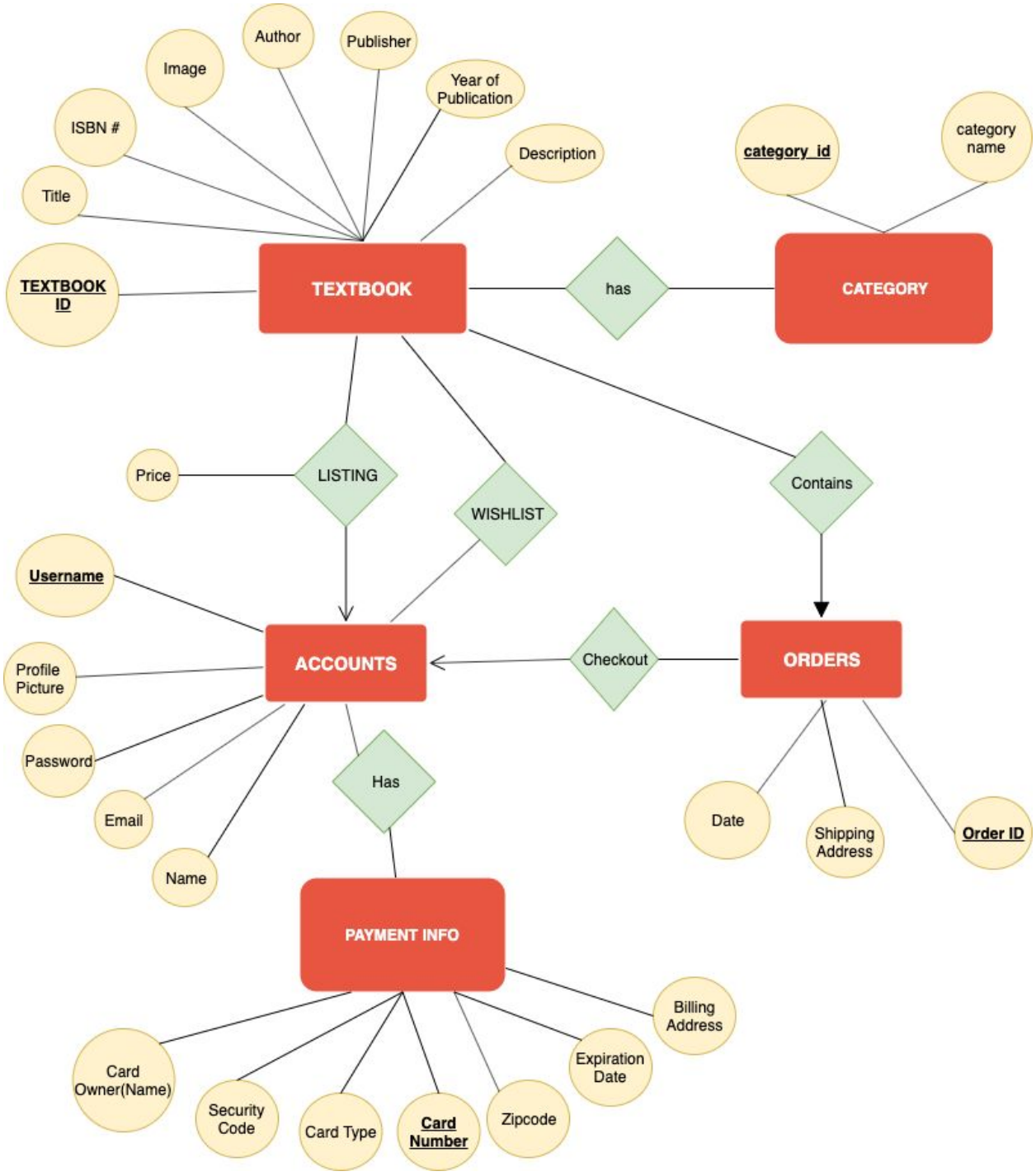
Project DB Model Design Revision

Team 25:
Au Tran
Benny Ooi
Aldrich Mangune


Author
Publisher
Image
Year of Publication
ISBN #
Description
Title
category_id
category name
TEXTBOOK ID
TEXTBOOK
has
CATEGORY
Price
LISTING
WISHLIST
Contains
Username
ACCOUNTS
Checkout
ORDERS
Profile Picture
Password
Email
Name
Has
Date
Shipping Address
Order ID
PAYMENT INFO
Billing Address
Card Owner(Name)
Security Code
Card Type
Card Number
Zipcode
Expiration Date

Entity Sets:

1) Textbook - Entity set representing a textbook. It contains attributes ISBN-13 #, Title, Image, Author, Publisher, Year of Publication, Description. It has an auto-increment primary key textbook_ID

2) Category - An entity set representing a category that a textbook belongs to. It has an auto-increment primary key category_id and attribute category name.

3) Orders - an entity set representing a transaction/order made (order histories). It contains auto-increment key Order ID, Date, and Shipping Address attribute.

4) Accounts - An entity set representing the account of the users of our application. It contains attributes Password, Email, Profile Picture, Name, and primary key Username

5) Payment Info is an entity set representing a credit/debit card users used to pay for transactions. It contains attributes Card Owner(Name), Security Code, Card Type, Zipcode, Expiration Date, Billing Address and primary key is the 16-digit Card Number.

Relationships:

1) Listing is a many-to-one relationship between Accounts and Textbooks. Each user would be able to add many textbook listings for sale and each textbook can only points to 1 account.
2) Wishlist is a many-to-many relationship between Accounts and Textbooks. Each user would be able to add many textbooks to their wishlist and each textbooks could be in many users' wishlist.
3) Checkout is a many-to-one relationship between Accounts and Order. The user would be able to checkout many orders while each order must be bind to 1 account.
4) Between Orders and Textbooks, there is a many-to-one relationship Contains where an order can contain multiple textbooks while each textbook can only belong to 1 order.
5) Between Textbook and Category, the many-to-many relationship Has represents that 1 category can have many textbooks and many textbooks can belong to many different categories.
6) Between Accounts and PaymentInfo, the many-to-many relationship Has represents that 1 account can have multiple payment methods and 1 credit card can be used for transactions in multiple accounts.

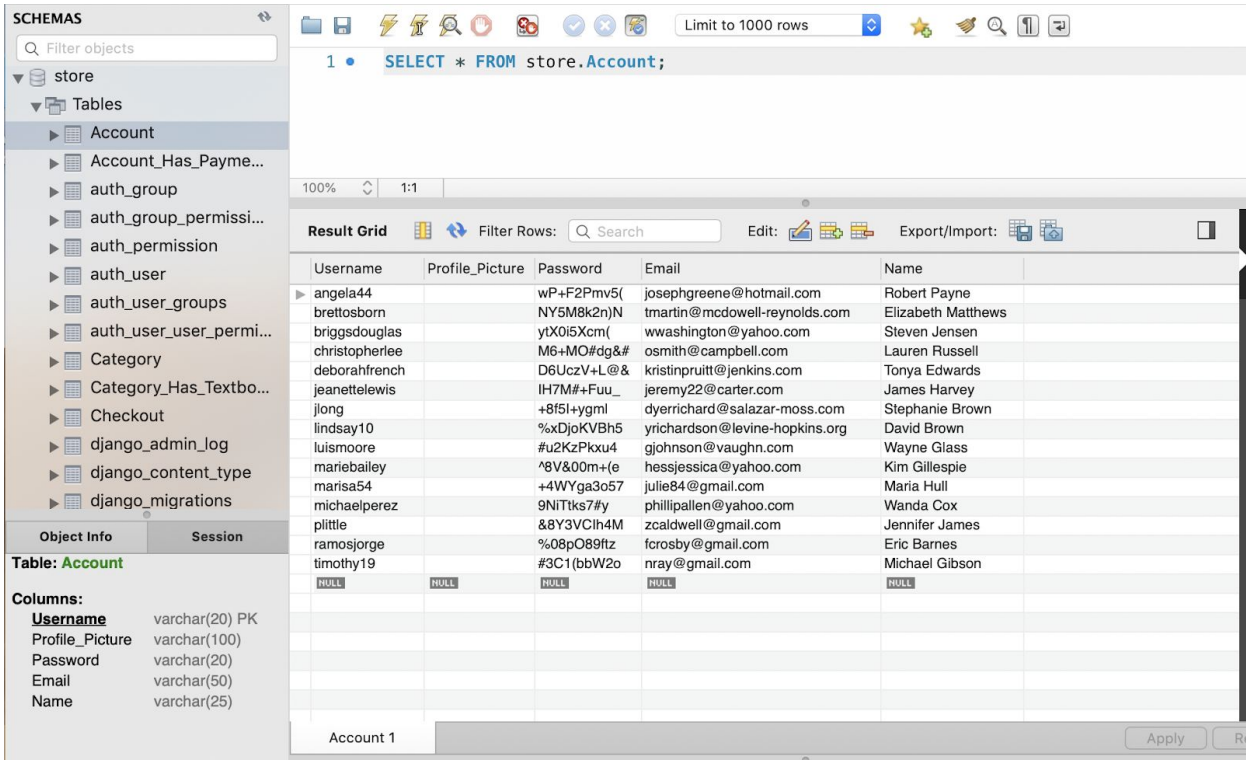
SCHEMA

We are using Django as part of our 3 tier architecture and unfortunately Django comes with an ORM that implements a database table as an object in Python. This mean that I have to specify my schemas as classes in Python and Django will automatically convert the classes into MySQL tables and each object instance I created will be converted to a new table instance in MySQL database.

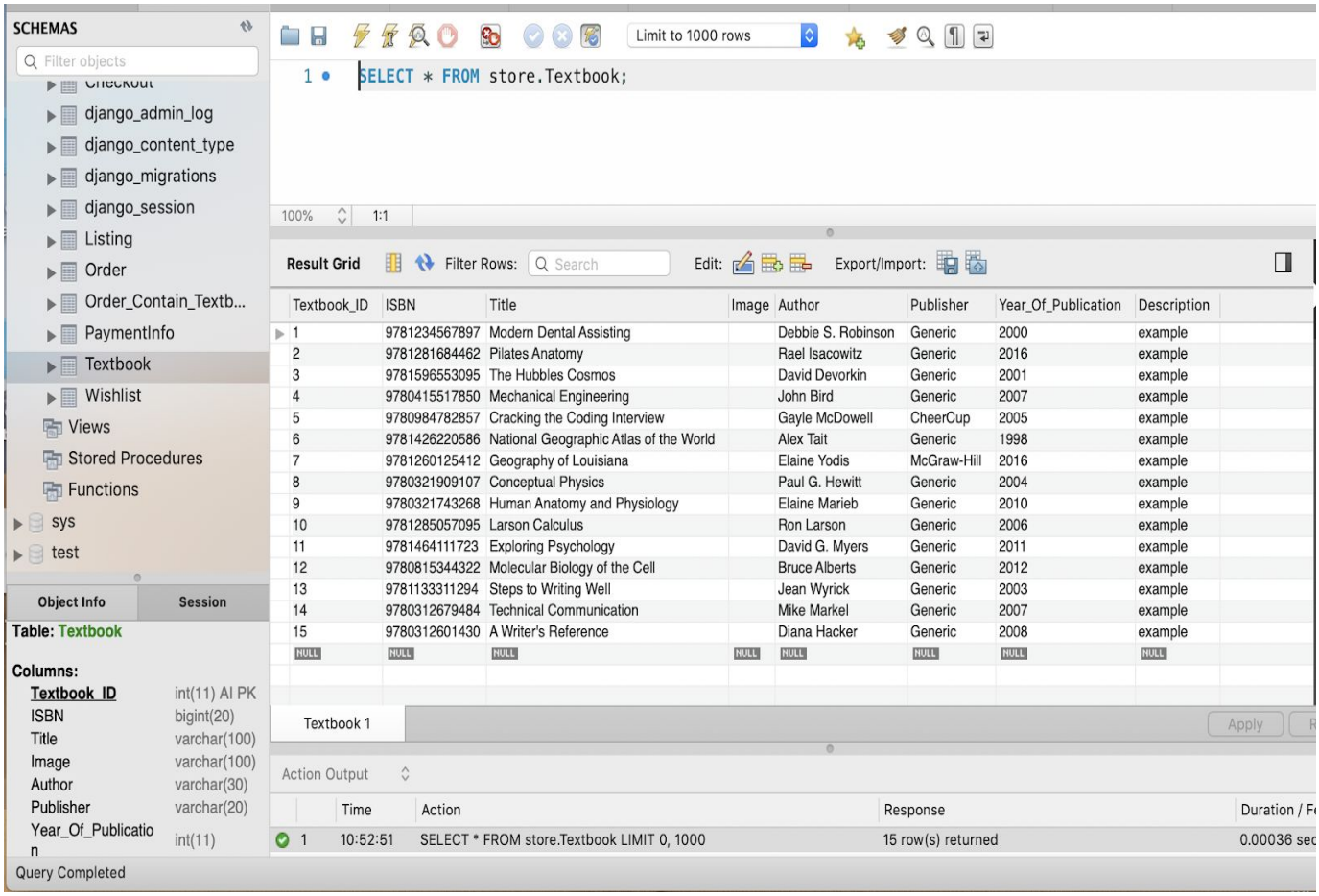
1.) The problem is that Django does not support the use of composite primary key. It requires that only 1 column can be primary key maximum. Therefore, in the tables that represent relationship Django automatically add a field id as the auto-increment primary key. For example, relationship Wishlist is a relationship that connects entity set Account and Textbook. The table for Wishlist will become Wishlist(**id,** Account_key, Textbook_key). However, Django lets me add a constraint specifying that no instances of Wishlist can have the same Account_key and Textbook_key so it behave the same as a primary key. The only setback is that those relationship will have a mandatory auto-increment primary key id, the behavior will be the same.

2.) Django automates the conversion of my schemas into Database tables. However, the naming convention can be a little disambiguous. For example, Wishlist again is a relationship connecting entity set Account and Textbook so the table will contain the key of these 2 attributes. Account have primary key **Username** and Textbook have primary key **Textbook_ID**. When Django convert this schema into a table, it will create a table Wishlist with column names Account_id and Textbook_id. This Account_id does actually contain the primary key **Username** so I hope it doesn't confuse you thinking its some other attribute.

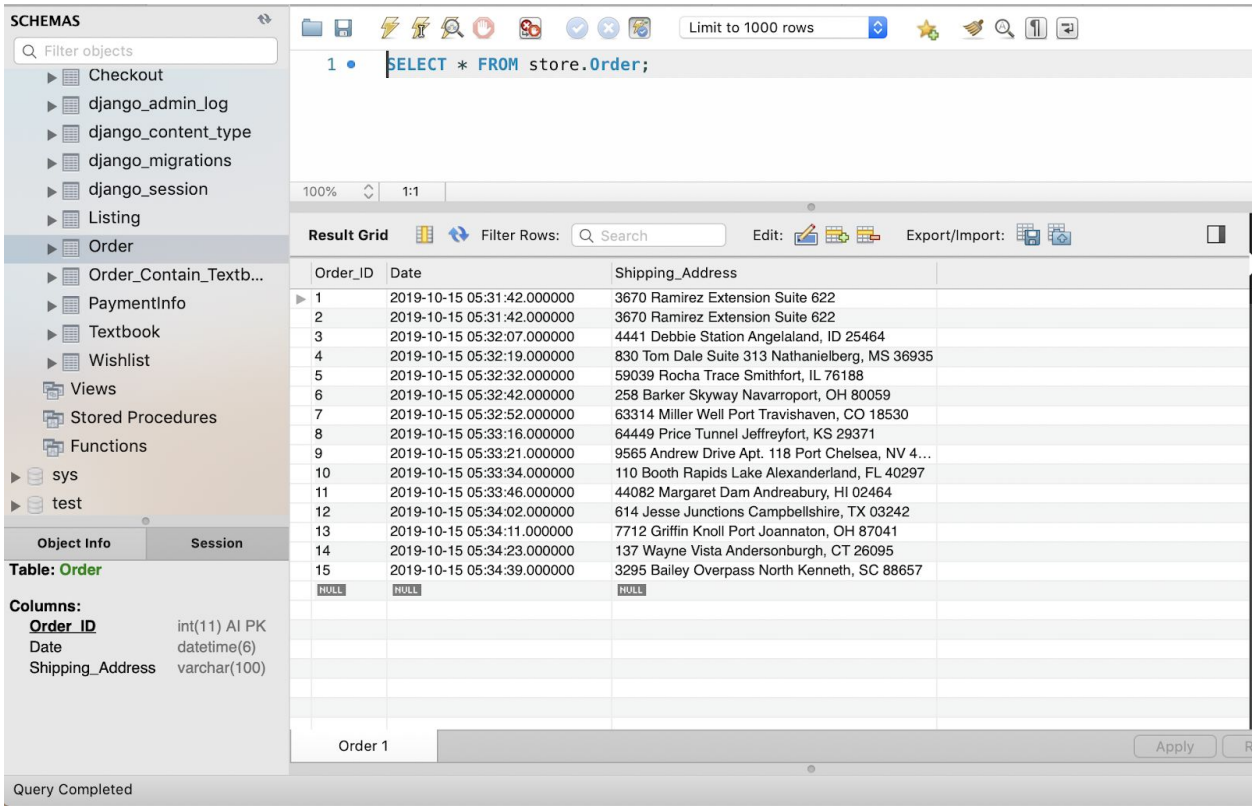
3.) Some of the table have an Image attribute that I didn't add to the table.



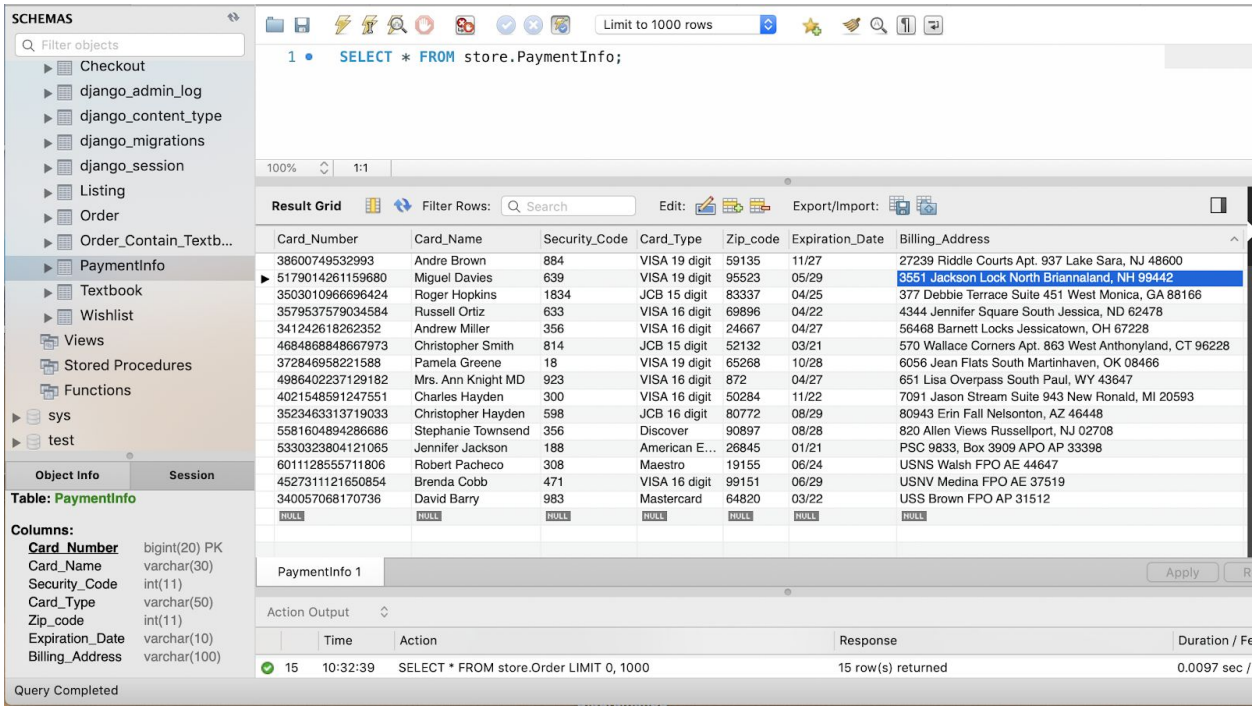
Accounts(**Username**, Profile_Picture, Password, Email, Name)



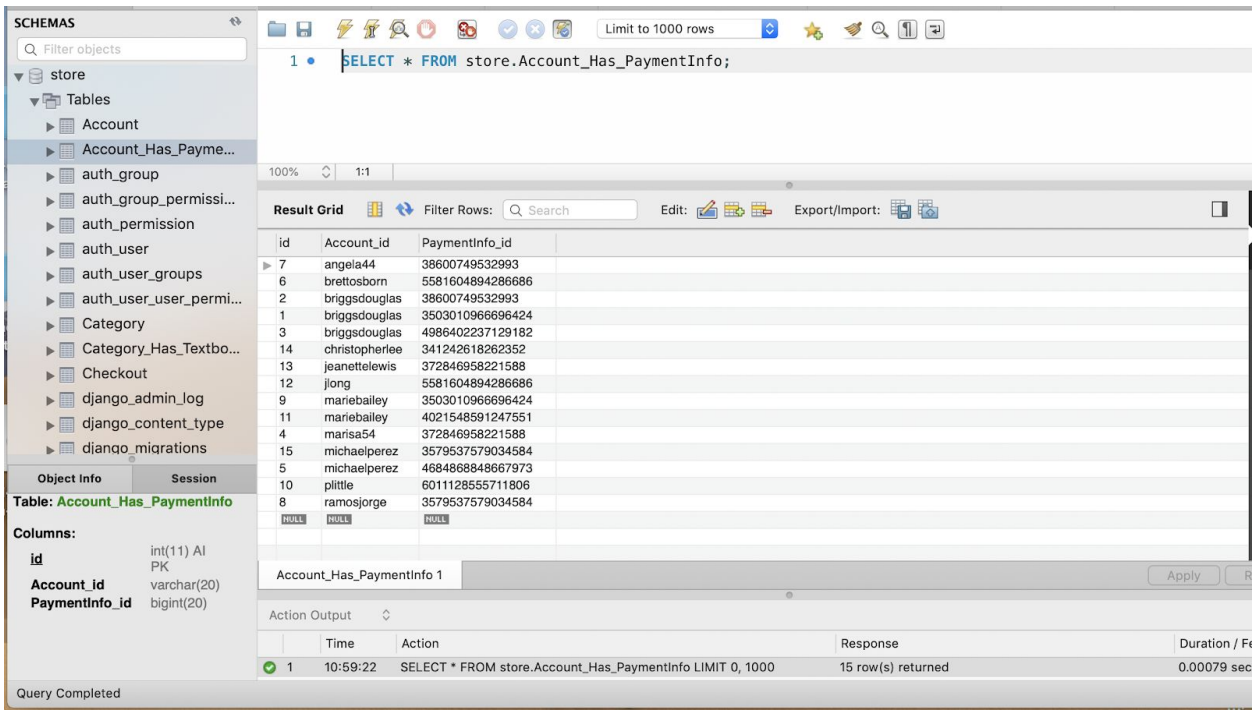
Textbook(**Textbook_ID**, ISBN, Title, Image, Author, Publisher, Year_Of_Publication, Description)



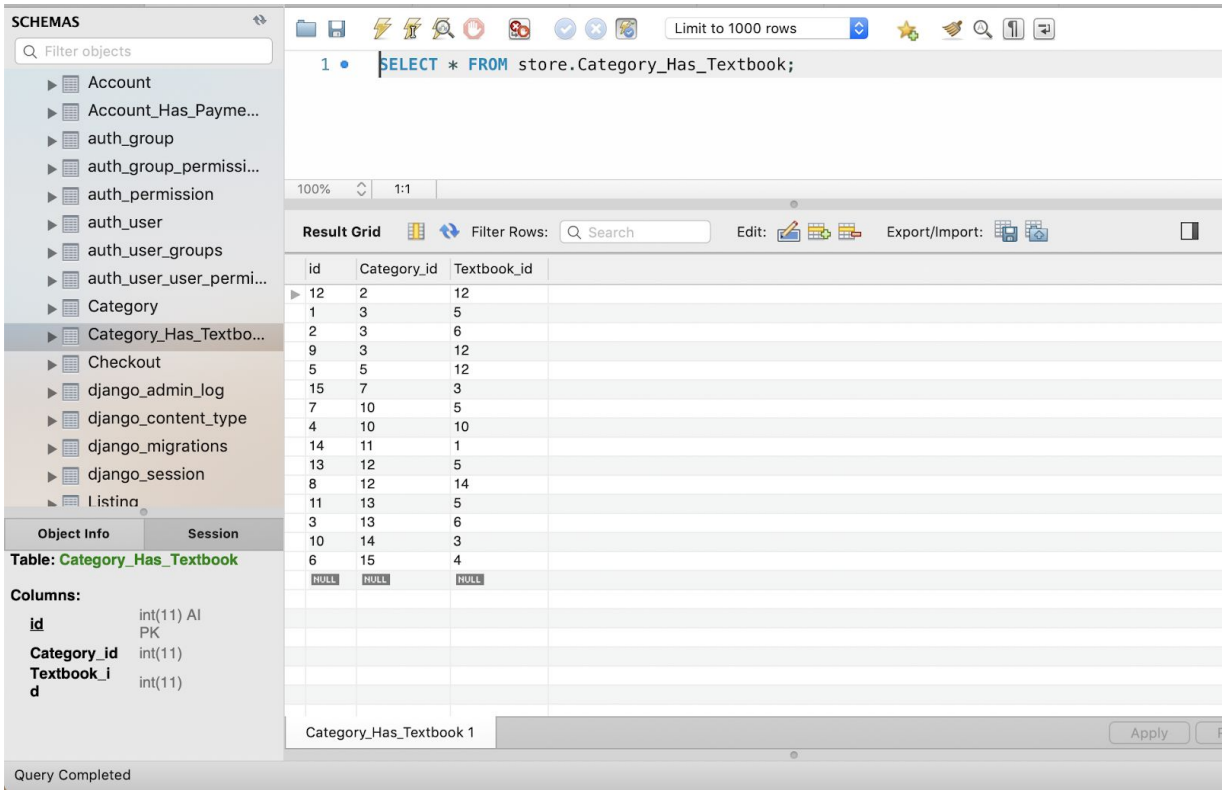
Order(**Order_ID**, Date, Shipping_Address)



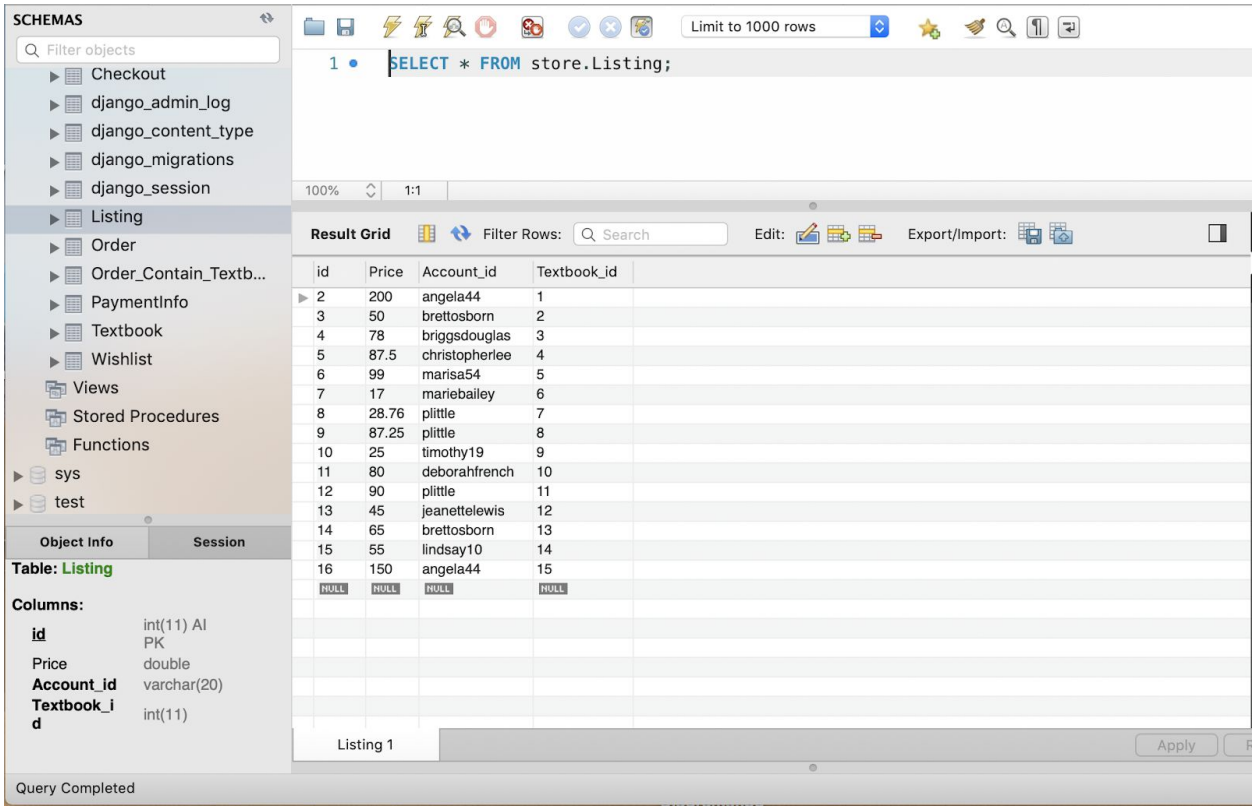
PaymentInfo(**Card Number**, Card Name, Security Code, Card Type, Zipcode, Expiration Date, Billing Address)



Account_Has_PaymentInfo(**id** Account_id, Paymentinfo_id)
The auto increment id primary key is required and automatically added by Django.
This Account_id is primary key of Account which is Username, and Paymentinfo_id is actually primary key of PaymentInfo which is Card Number.



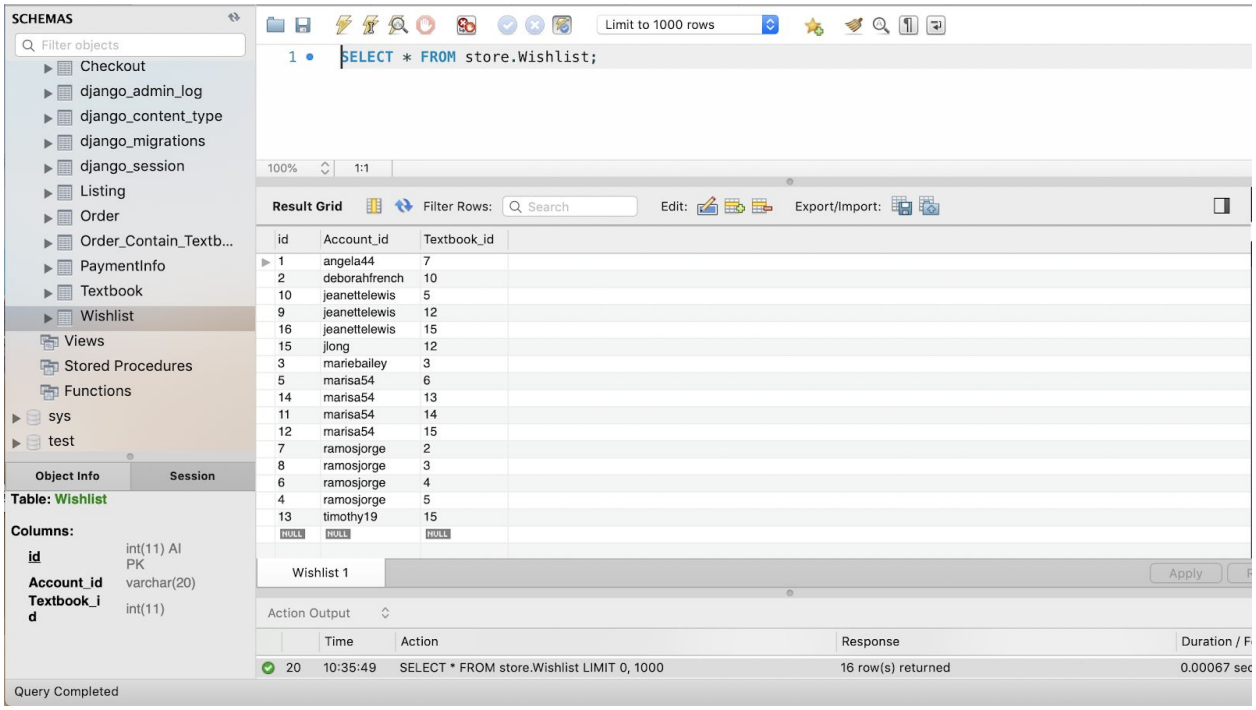
Category_Has_Textbook(**id**, Category_id, Textbook_id)
The auto increment id primary key is required and automatically added by Django.



Listing(**id**, Price, Account_id, Textbook_id)

The auto increment id primary key is required and automatically added by Django.

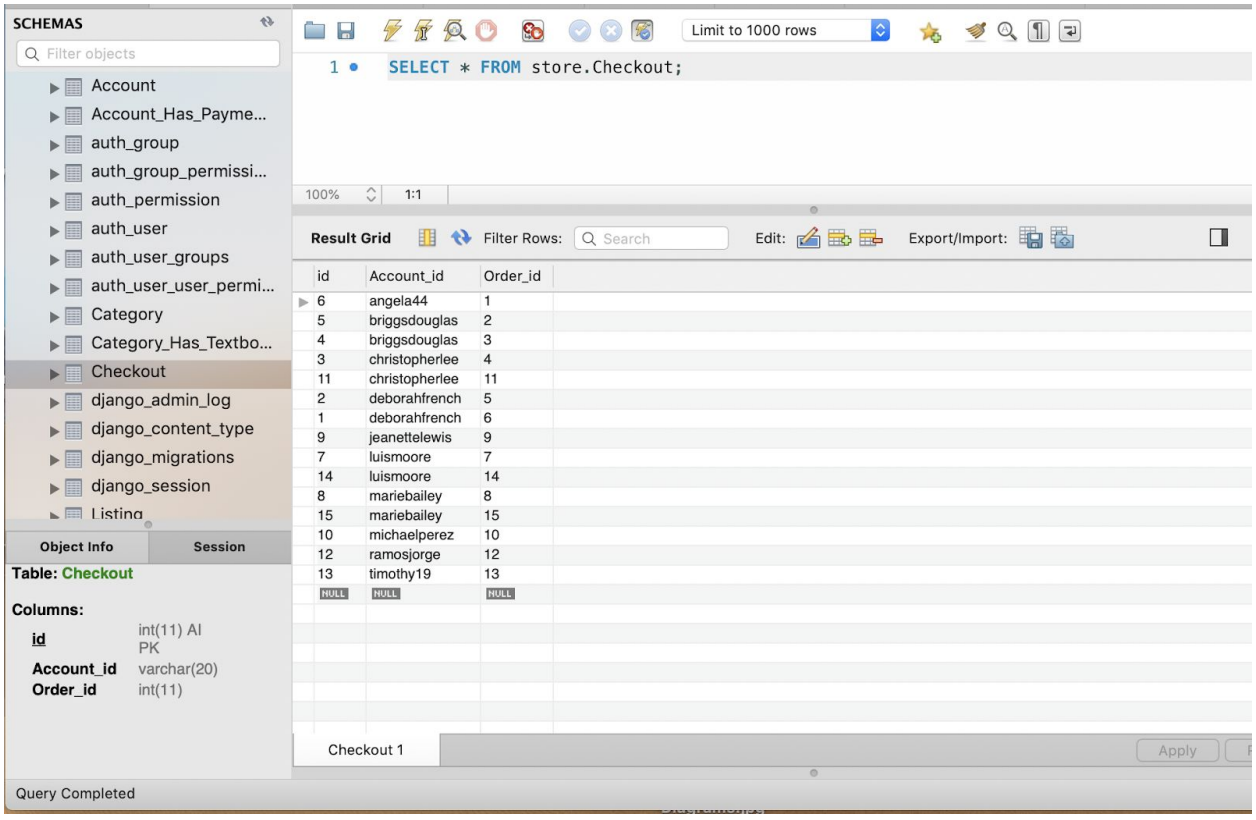
This Account_id is primary key of Account which is Username



Wishlist(**id**, Account_id, Textbook_id)

The auto increment id primary key is required and automatically added by Django.

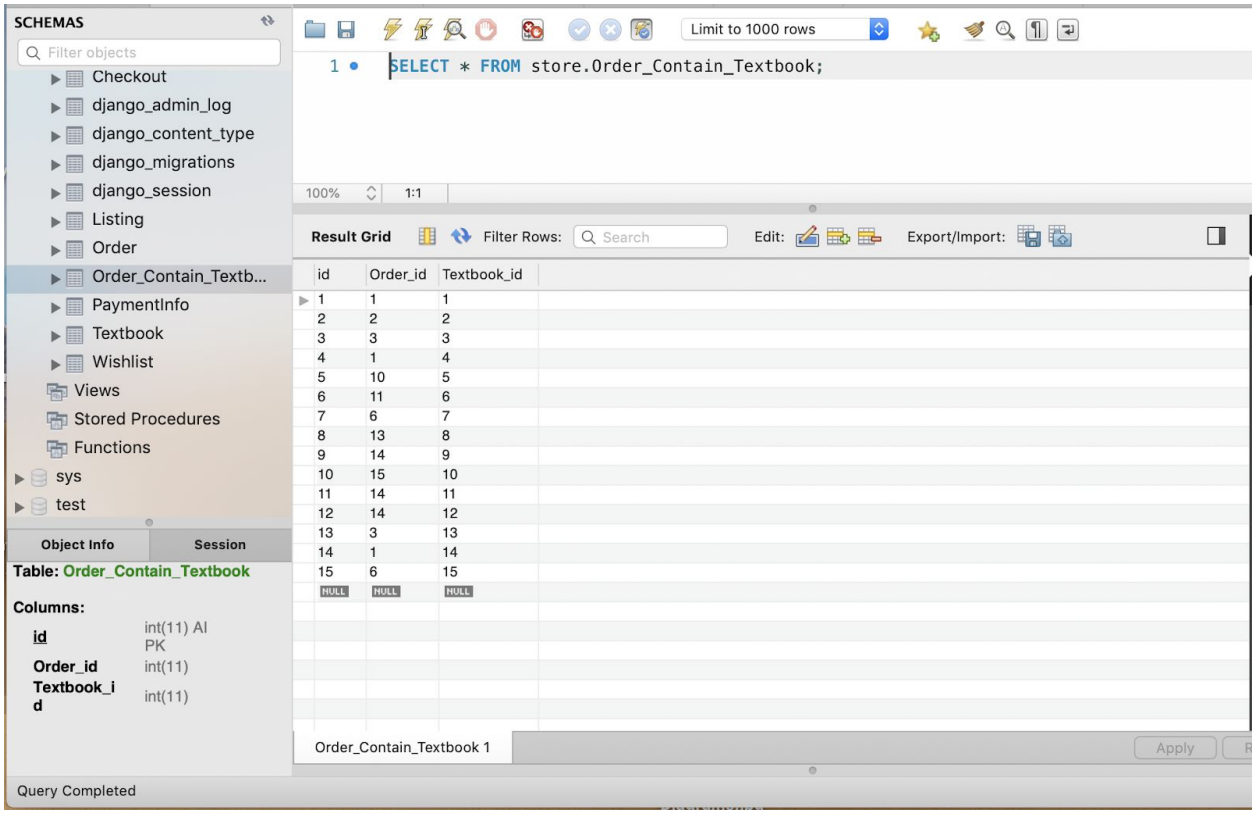
This Account_id is primary key of Account which is Username



Checkout(**id**, Account_id, Order_id)

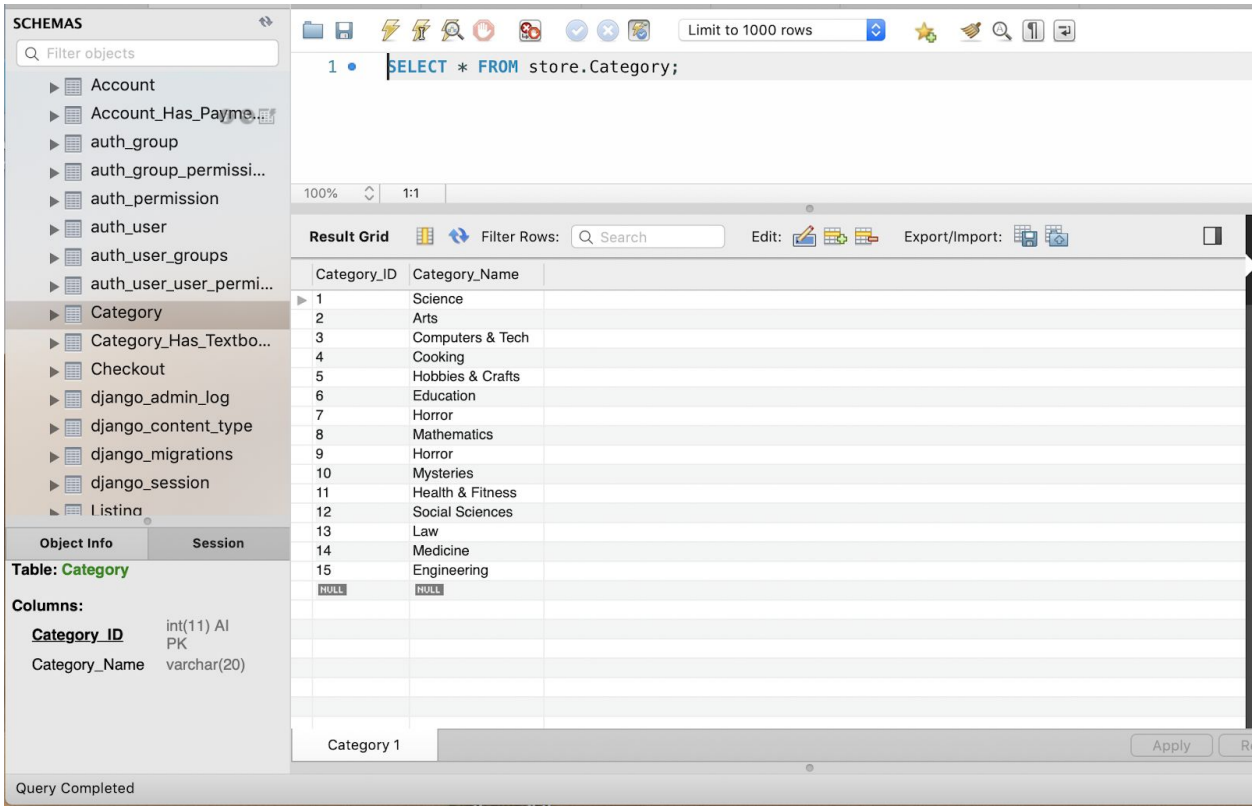
The auto increment id primary key is required and automatically added by Django.

This Account_id is primary key of Account which is Username.



Order_Contain_Textbook(**id**, Order_id, Textbook_id)

The auto increment id primary key is required and automatically added by Django.



Category(**Category_ID,** Category_Name)