**Introduction to**

**Database Design & Development (IDDD)**

Name: Nguyen Thanh Son

CTNumber: CT0338434

Class: FTDip2104\_IT69

Lecturer: MeiLing Tan

Assignment Due Date: Wednesday, 11 August 2021, 11:55 PM

Table of Contents

[Question 1 3](#_Toc79569243)

[Question 2 5](#_Toc79569244)

[i) Table in UNF: 5](#_Toc79569245)

[ii) Table from UNF to 1NF: 6](#_Toc79569246)

[iv) Table in 3NF 7](#_Toc79569247)

[Question 3 9](#_Toc79569248)

[Customer table 10](#_Toc79569249)

[Product table 10](#_Toc79569250)

[OrderQuantity table 11](#_Toc79569251)

[Order Table 11](#_Toc79569252)

[Insert values into Customer table 12](#_Toc79569253)

[Insert values into Product table 13](#_Toc79569254)

[Insert values into OrderQuantity table 14](#_Toc79569255)

[Insert values into OrderTable table 15](#_Toc79569256)

[Projection 16](#_Toc79569257)

[Restriction 16](#_Toc79569258)

[Aliases 17](#_Toc79569259)

[NULL value handling 17](#_Toc79569260)

[Concatenation 18](#_Toc79569261)

[Comparison Operator 18](#_Toc79569262)

[Logical Operator 19](#_Toc79569263)

[Sorting 19](#_Toc79569264)

[Function () Group function & Single Row Function 20](#_Toc79569265)

[Joins and subqueries 22](#_Toc79569266)

Question 1 **(30 Marks)**

Consider the following set of requirements for a UNIVERSITY database that is used to keep track of student’s transcripts.

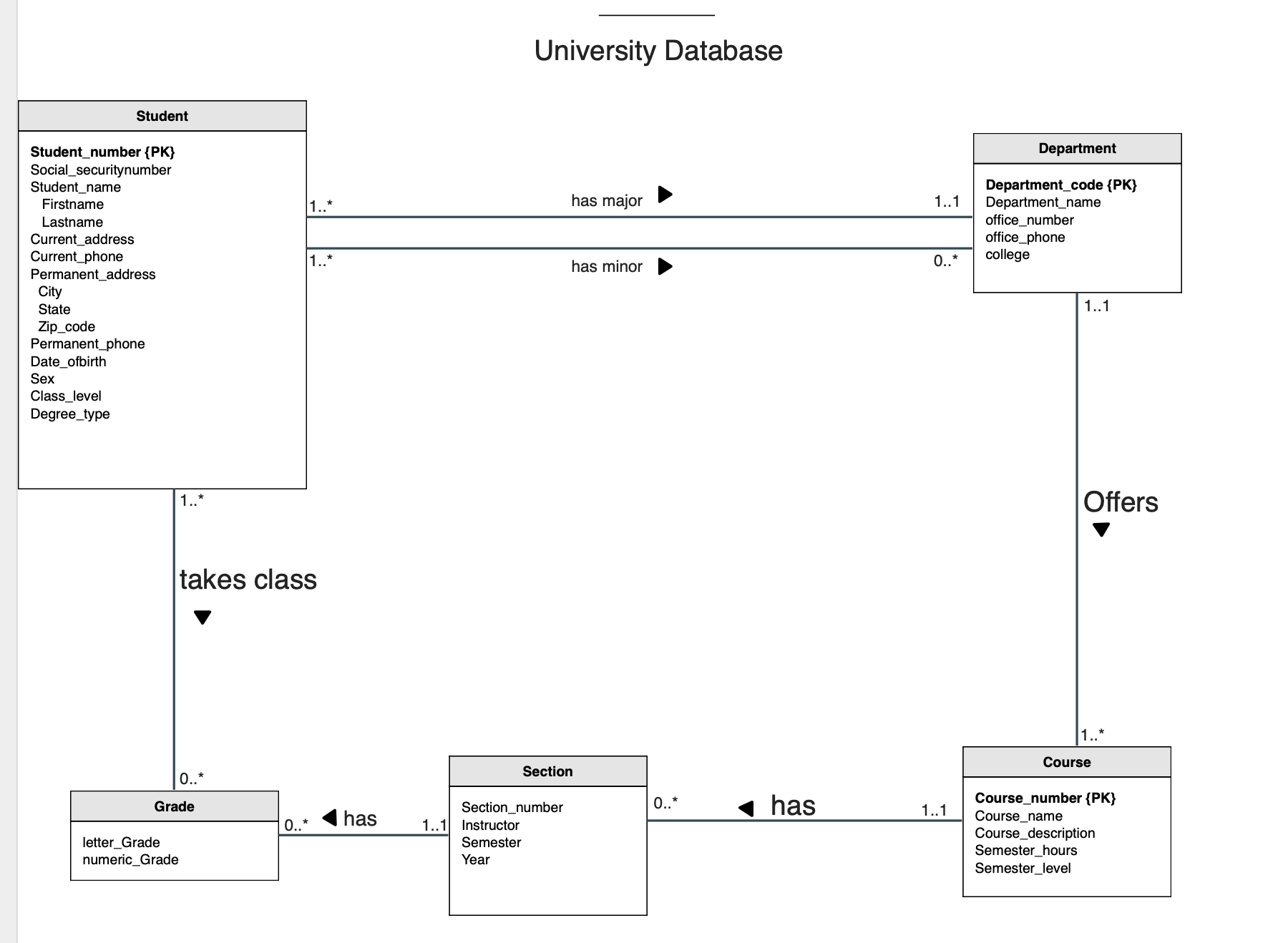
The university keeps track of each student’s name, student number, Social Security number, current address and phone number, permanent address and phone number, birth date, sex, class (freshman, sophomore, ..., graduate), major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and ZIP Code of the student's permanent address and to the students the last name. Both Social Security number and student number have unique values for each student.

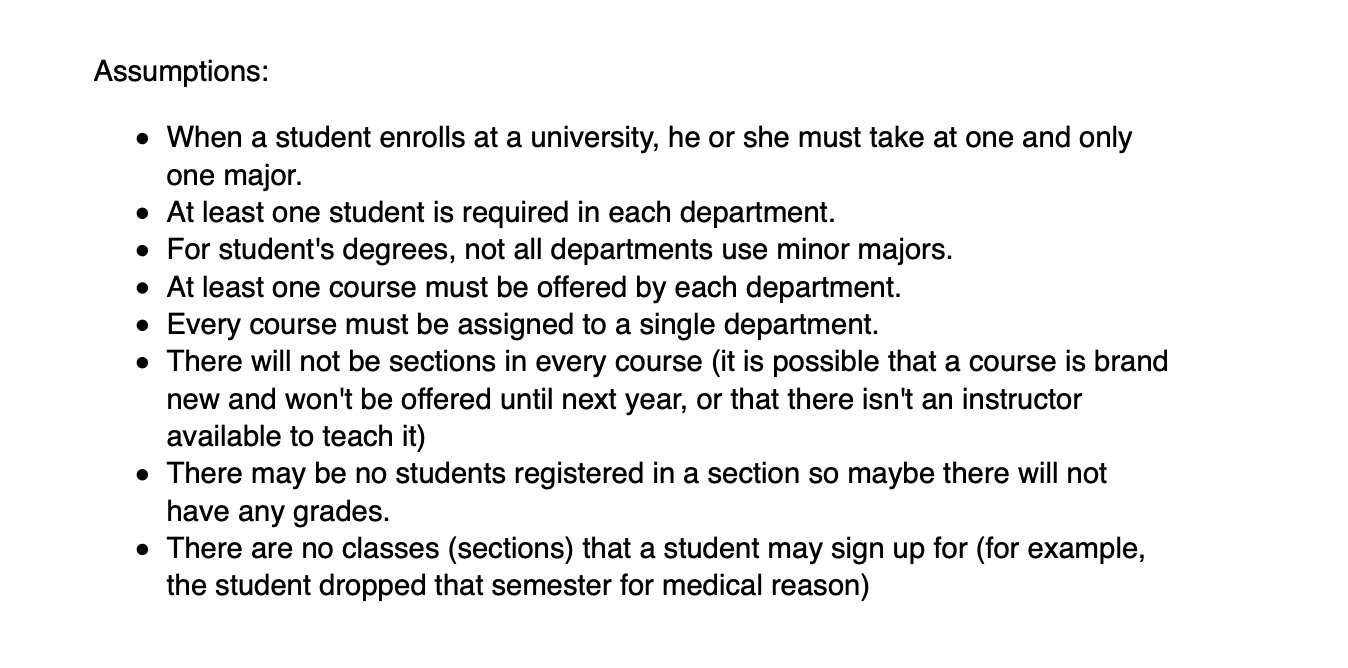
Each department is described by a name, department code, office number, office phone number, and college. Both name and code have unique values for each department.

Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of the course number is unique for each course.

Each section has an instructor, semester, year, course, and section number. The section number distinguishes sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.

A grade report has a student, section, letter grade, and numeric grade (0, 1, 2, 3, or 4). Create an ER Diagram, listing all cardinality constraints and assumptions.

****

****

Legends: 1..\* = One to Many

0…\* = Zero to many

1..1 = One to One

Question 2**:** ****

Abbreviations:

Order Number = OrderNo

Customer Number = CustNo

Customer Name = CustNa

Customer Address = CustAdd

City-Country = C-C

Unit Price = U-P

## i) Table in UNF:

**OrderForm** (OrderNo, CustNo, CustNa, CustAdd, Date, {City-Country, ProductNo, Description, Quantity, UnitPrice})

{ City-Country, ProductNo, Description, Quantity, UnitPrice} => UNF because there are multiple values under them.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OrderNo | Date | CustNo | CustNa | CustAdd | C-C | ProductNo | Description | Quantity | U-P |
| 1234 | 11/04/98 | 9876 | Billy | 456 HighTower St | HongKong, China | A123  B234  C345 | Pencil  Eraser  Sharpener | 100  200  5 | $3.00  $1.50  $8.00 |

**Assumptions:** One customer only orders a type of product once.

ii) Table from UNF to 1NF: In this step, i removed any **duplicated attributes** and **repeating groups** by transferring the entire contents of that group with a new row. ProductNo, Description, Quantity, and UnitPrice information has been divided into their own rows. The table below is now in first normal form.

**OrderForm** (OrderNo, CustNo, CustNa, CustAdd, City, Country, Date, ProductNo, Desciption, Quantity, UnitPrice)

Primary Key: **OrderNo** and **ProductNo**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OrderNo** | Date | CustNo | CustNa | CustAdd | City | Country | **ProNo** | Description | Quantity | U-P |
| 1234 | 11/04/98 | 9876 | Billy | 456 HighTower St | HongKong | China | A123 | Pencil | 100 | $3.00 |
| 1234 | 11/04/98 | 9876 | Billy | 456 HighTower St | HongKong | China | B234 | Eraser | 200 | $1.50 |
| 1234 | 11/04/98 | 9876 | Billy | 456 HighTower St | HongKong | China | C345 | Sharpener | 5 | $8.00 |

iii) Table in 2NF: In the second normal form, all partial functional dependencies have been removed. These are attributes that depend directly on more than one primary key. I already listed the dependencies under that Quantity is dependent on both ProductNumber and OrderNumber. For this reason, this is a full dependency. Next, i put both OrderNumber and ProductNumber along with Quantity into a separated table (make up the composite key).

**OrderNo, ProductNo** —> Quantity (Quantity has to depend on both ProductNo and OrderNo because only ProductNo as Primary Key cannot define how many quantity that product has and a ProductNo could have many quantity) => full dependency

Partial dependencies:

**OrderNo** —> Date, CustNo, CustNa, CustAdd, City, Country (non-key depends on part of Primary Key)

**ProductNo** —> Description, UnitPrice (Non-key depends on part of Primary Key)

Then i divided the remaining values into two more tables for **OrderNumber** and **ProductNumber** with their dependencies.

**OrderTable** (OrderNo, Date, CustNo, CustNa, CustAdd, City, Country)

**Product** (ProductNo, Description, UnitPrice)

**OrderQuantity** (ProductNumber, OrderNumber, Quantity)

*Foreign Key****: ProductNumber*** *references Product (ProductNumber)*

*Foreign Key****: OrderNumber*** *references Order (OrderNumber)*

**OrderTable**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| OrderNo | Date | CustNo | CustNa | CustAdd | City | Country |
| 1234 | 11/04/98 | 9876 | Billy | 456 HighTower St | HongKong | China |

**Product**

|  |  |  |
| --- | --- | --- |
| ProductNo | Desctiption | Unit Price |
| A123 | Pencil | $3.00 |
| B234 | Eraser | $1.50 |
| C345 | Sharpener | $8.00 |

**OrderQuantity**

|  |  |  |
| --- | --- | --- |
| OrderNo | ProductNo | Quantity |
| 1234 | A123 | 100 |
| 1234 | B234 | 200 |
| 1234 | C345 | 5 |

iv) Table in 3NF: In the third normal form, all transitive dependencies must be removed. This happens when a non-primary key such as CustomerName, CustomerAddress and City, Country are dependent upon more than one key (listed in below). Hence, CustNo is still dependent on OrderNo so it becomes a foreign key in the Order table.

Transitive dependency:

CustNo ———> CustomerName, CustomerAddress, City, Country  (this is a non-key that depends on another non-key) => transitive dependency

OrderNo ———> Date, CustNo

**Customer** (CustNo, CustNa, CustAdd, City, Country)

**OrderTable** (OrderNo, Date, CustNo)

*Foreign Key:* ***CustNo*** *references Customer(CustNo)*

**Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CustNo | CustNa | CustAdd | City | Country |
| 9876 | Billy | 456 HighTower St | HongKong | China |

**OrderTable**

|  |  |  |
| --- | --- | --- |
| OrderNo | Date | CustNo |
| 1234 | 11/04/98 | 9876 |

The table is finally normalised from UNF into 3NF with 4 separate tables.

**Customer** (CustNo, CustNa, CustAdd, City, Country)

**Product** (ProductNo, Description, UnitPrice)

**OrderQuantity** (OrderNo, ProductNo, Quantity)

**OrderTable** (OrderNo, Date, CustNo)

**Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CustNo | CustNa | CustAdd | City | Country |
| 9876 | Billy | 456 HighTower St | HongKong | China |

**Product**

|  |  |  |
| --- | --- | --- |
| ProductNo | Desctiption | Unit Price |
| A123 | Pencil | $3.00 |
| B234 | Eraser | $1.50 |
| C345 | Sharpener | $8.00 |

**OrderQuantity**

|  |  |  |
| --- | --- | --- |
| OrderNo | ProductNo | Quantity |
| 1234 | A123 | 100 |
| 1234 | B234 | 200 |
| 1234 | C345 | 5 |

**OrderTable**

|  |  |  |
| --- | --- | --- |
| OrderNo | Date | CustNo |
| 1234 | 11/04/98 | 9876 |

Question 3 **(35 Marks)**

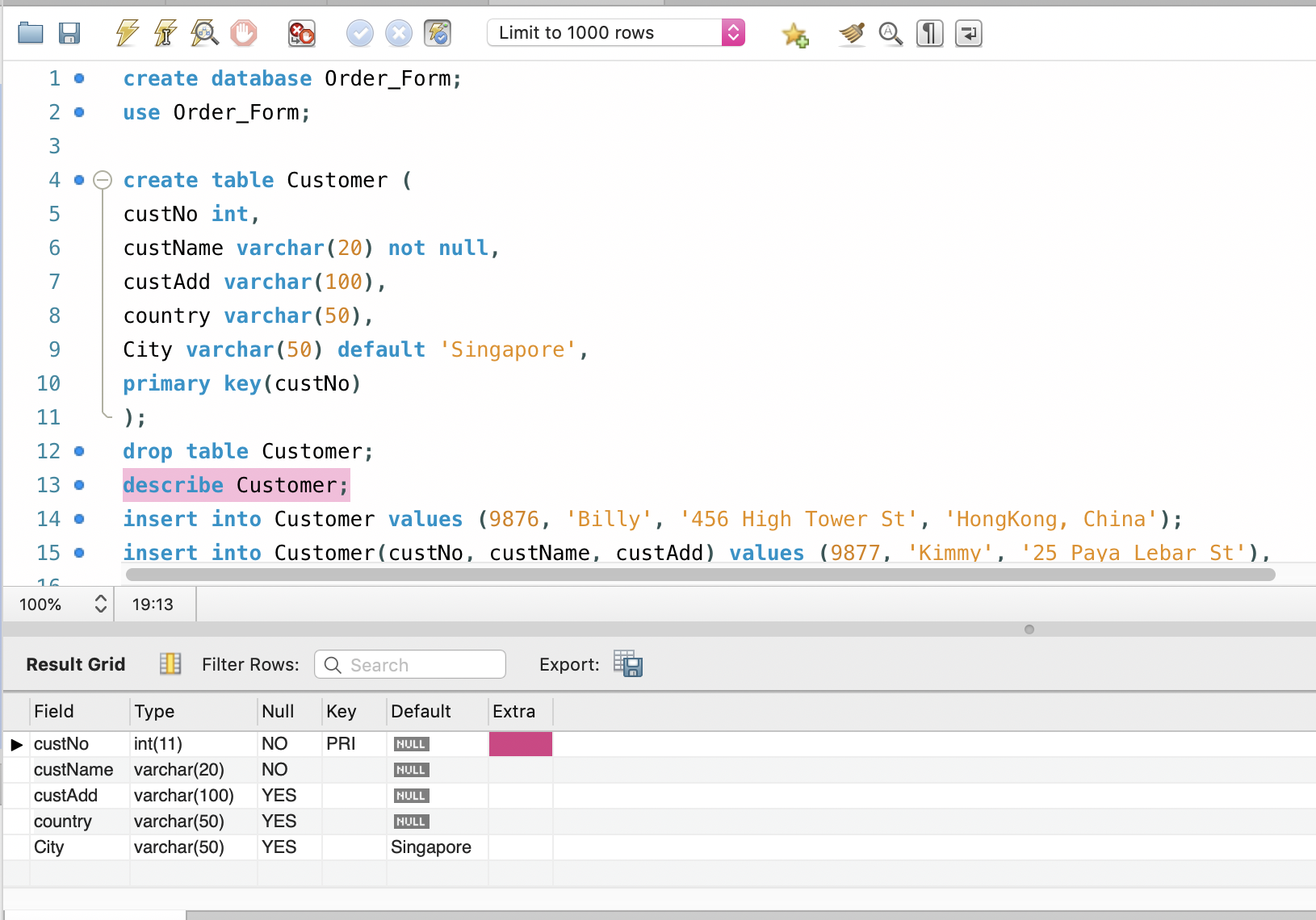
The physical database can be created using DDL (Data Definition Language) and manipulated using DML (Data Manipulation Language), which enable users to access or manipulate data as organized by the appropriate model.

1. i)  Using the CREATE TABLE in any dialect of SQL, show the corresponding SQL tables. Indicate the primary key, foreign keys, column names, constraints etc.
2. ii)  Insert (at least) 10 sample rows for each identified table.

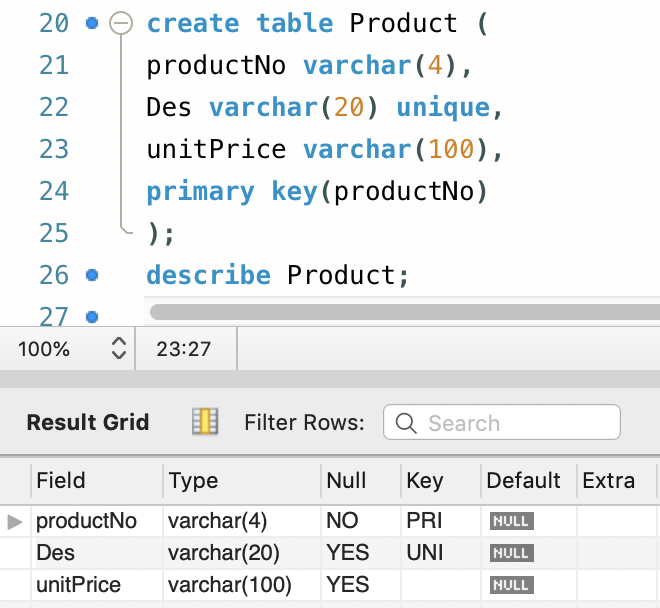
Student must test their database using at least the following:

* -  Restriction and Projection
* -  Aliases
* -  NULL value handling
* -  Concatenation
* -  Comparison Operator
* -  Logical Operator
* -  Sorting
* -  Function () Group function & Single Row Function
* -  Joins
* -  Subqueries

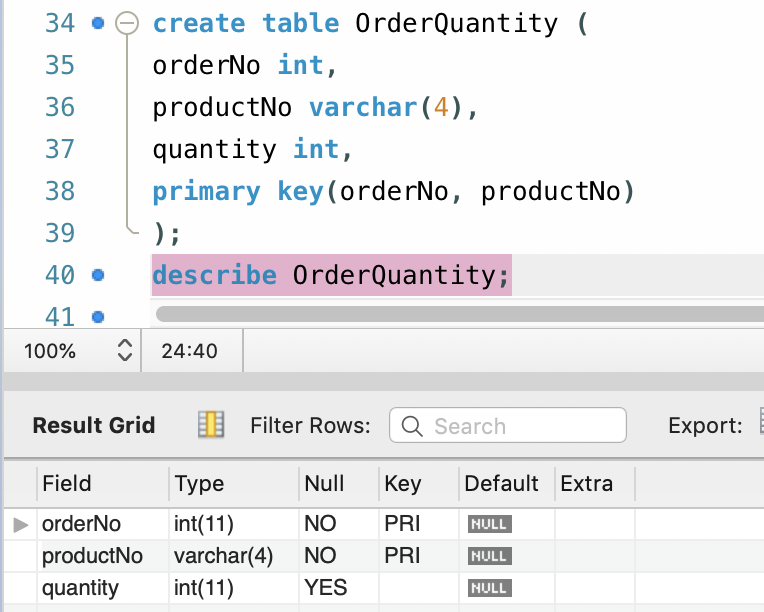
## Customer table

****

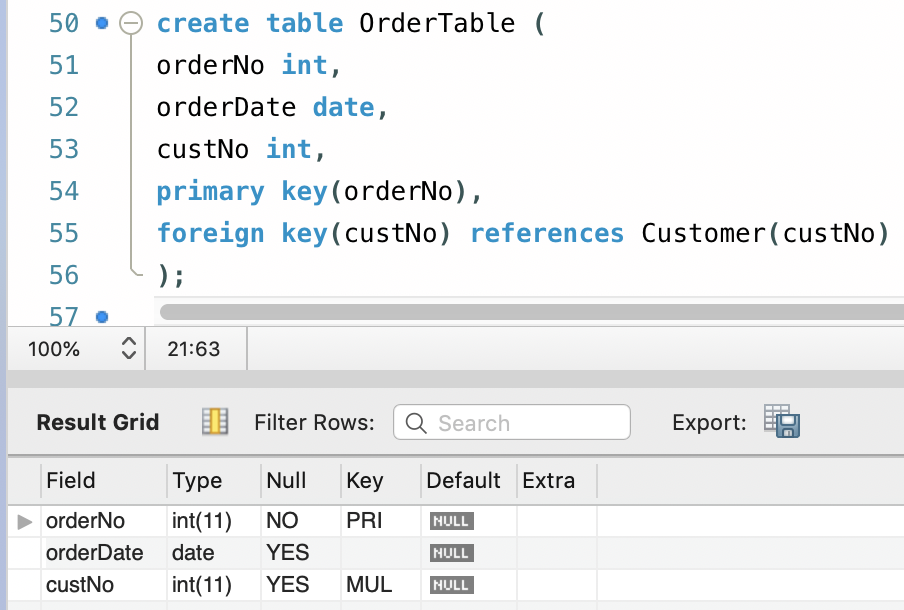
## Product table

****

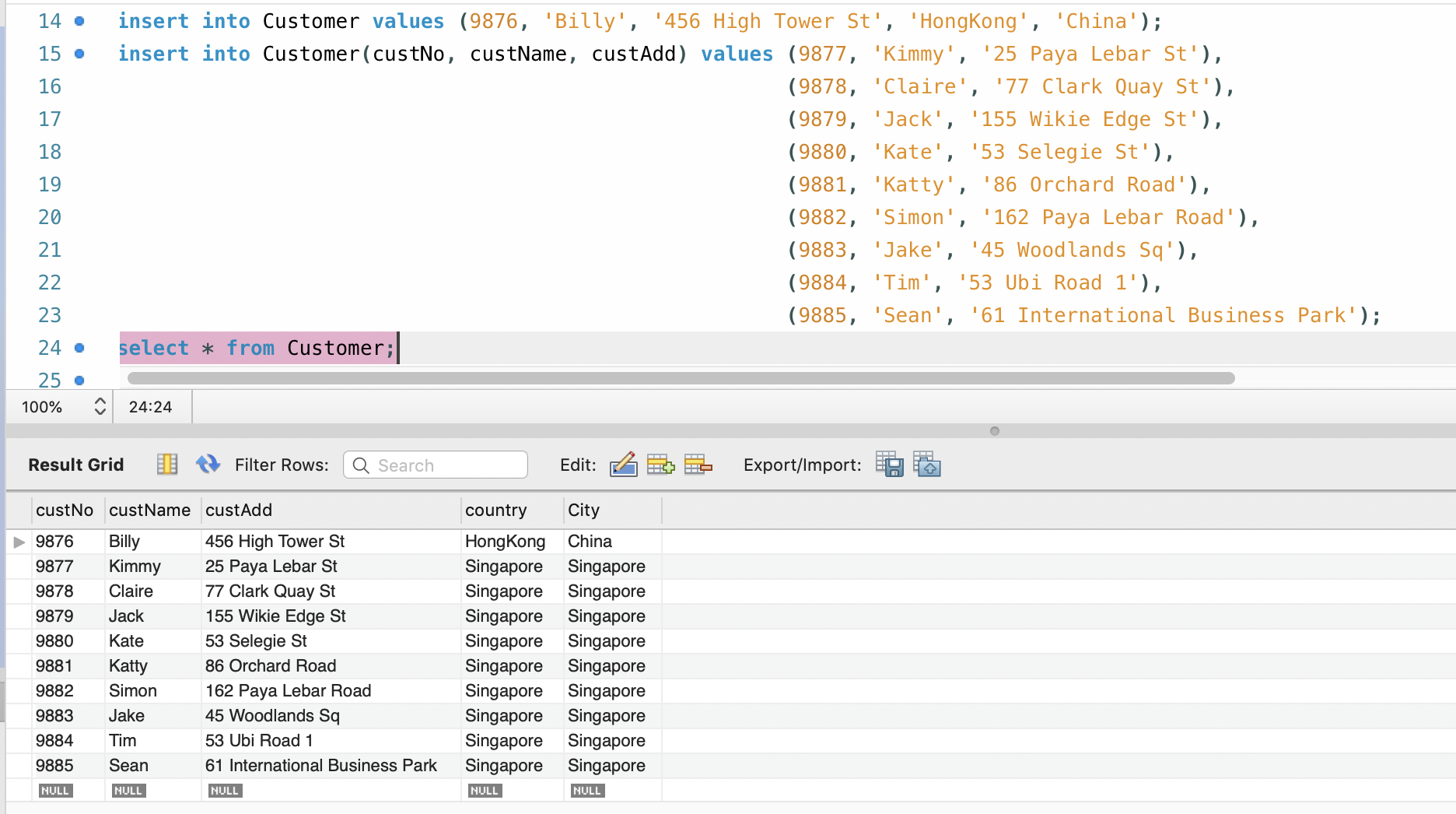
## OrderQuantity table

****

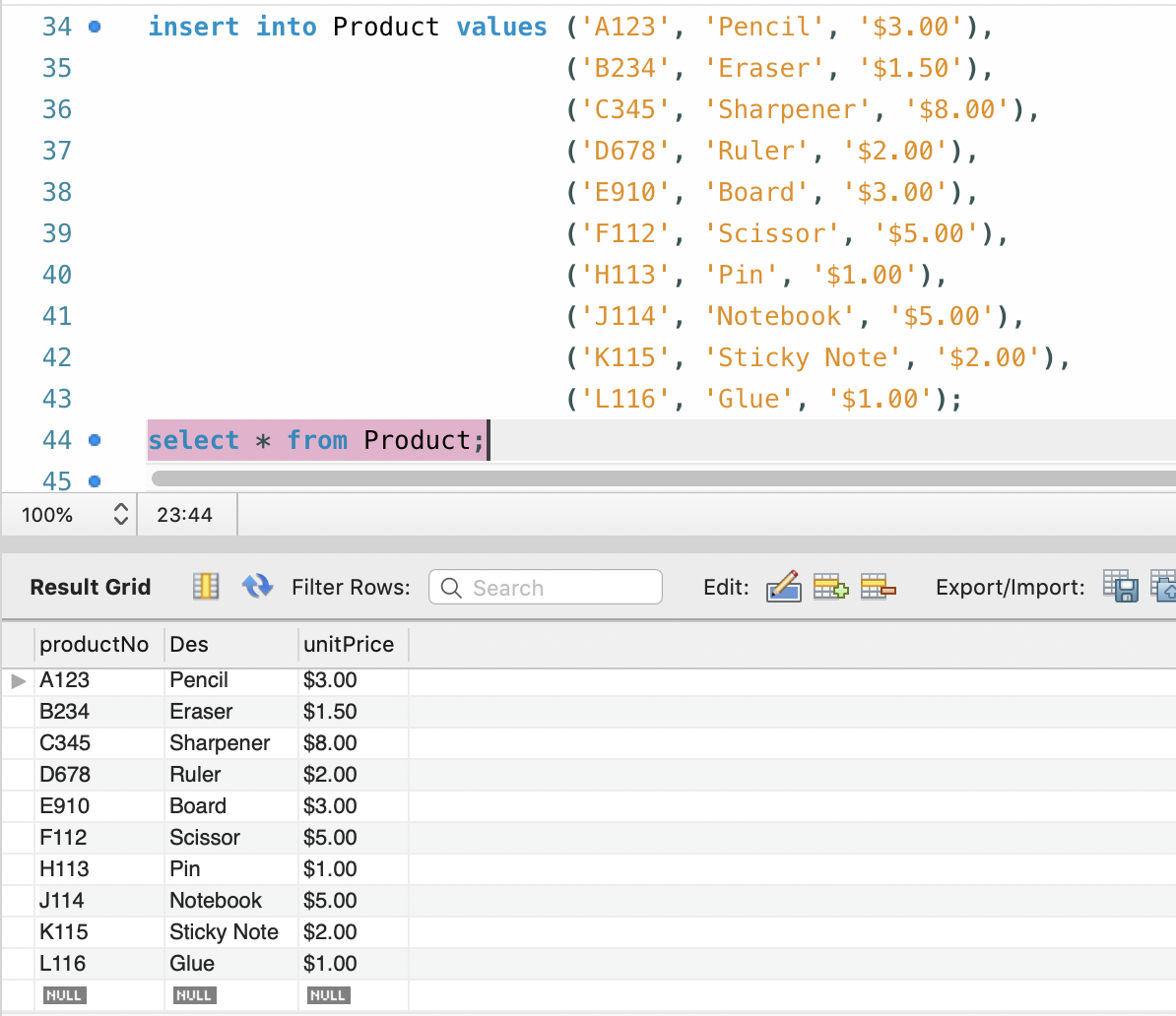
## Order Table

****

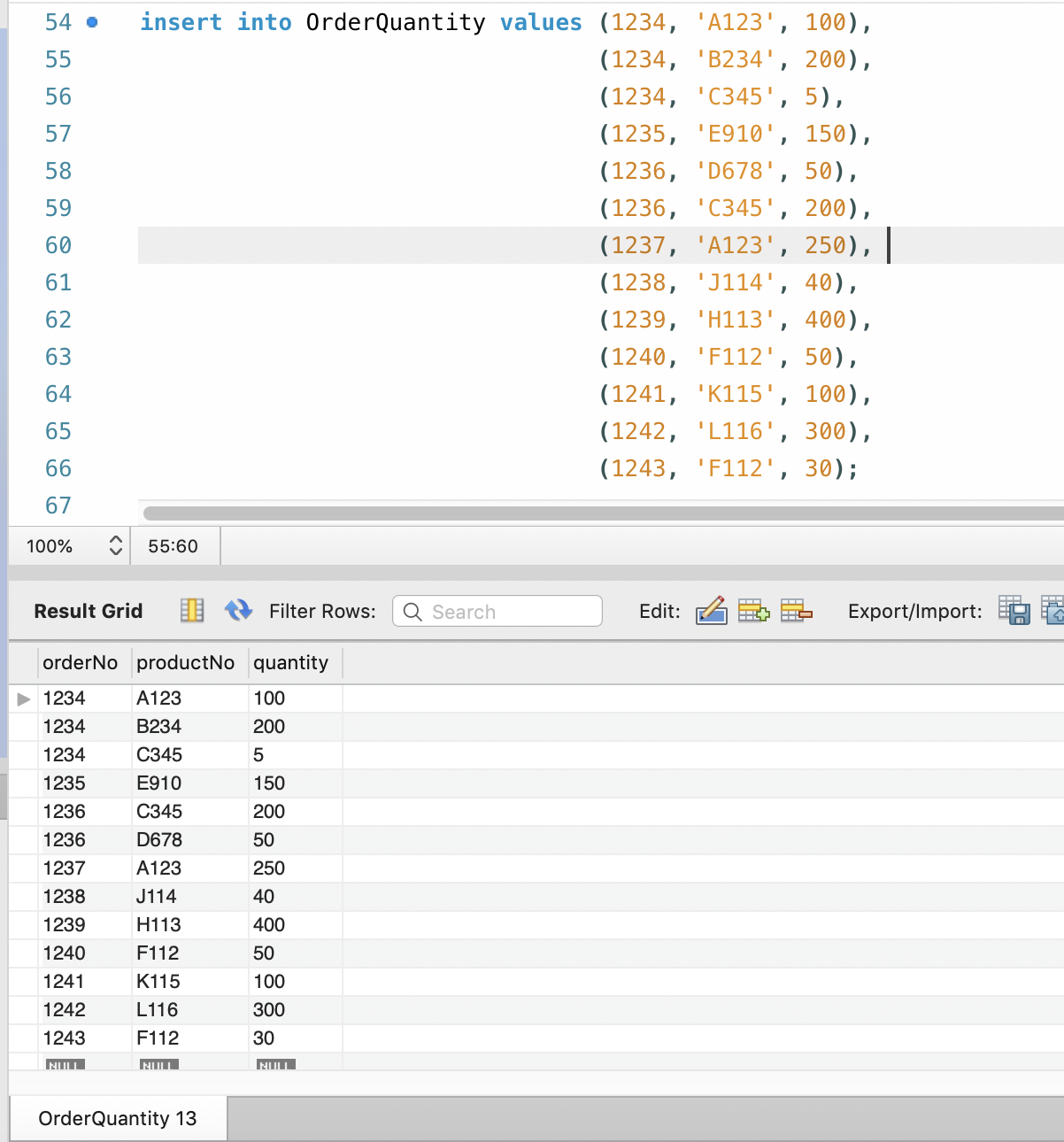
## Insert values into Customer table



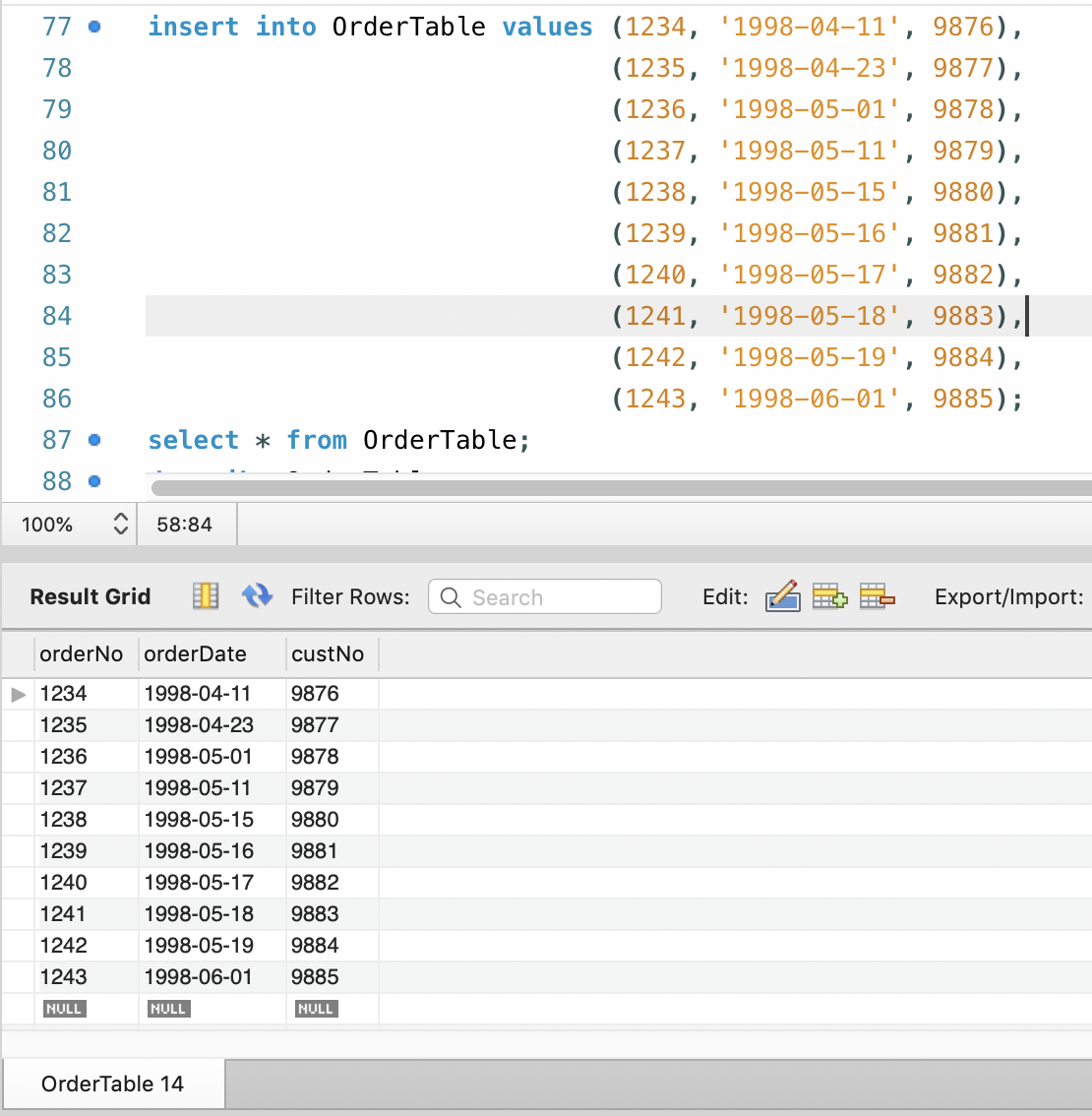
## Insert values into Product table



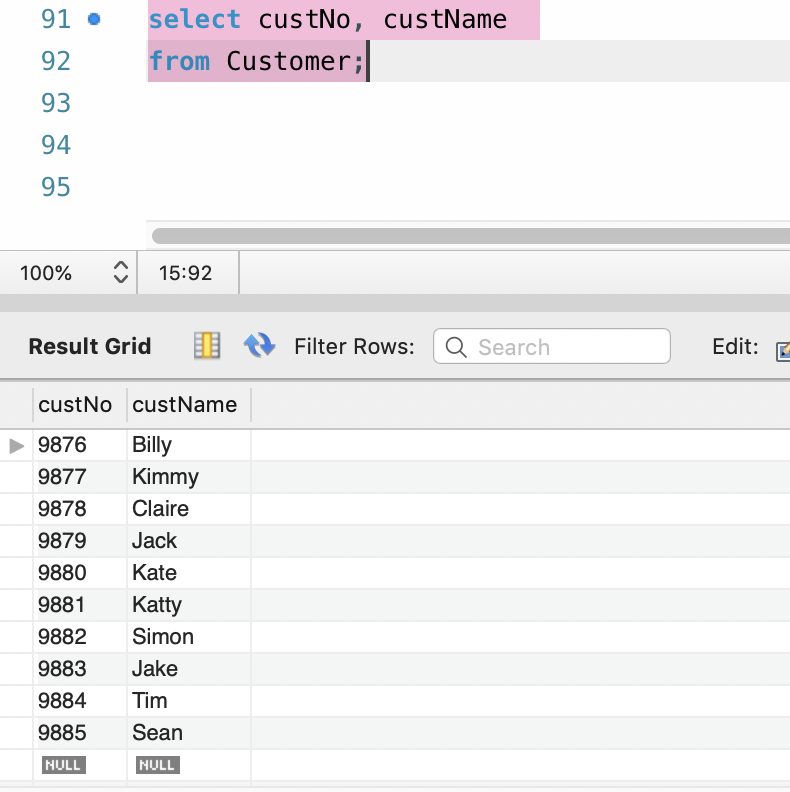
## Insert values into OrderQuantity table



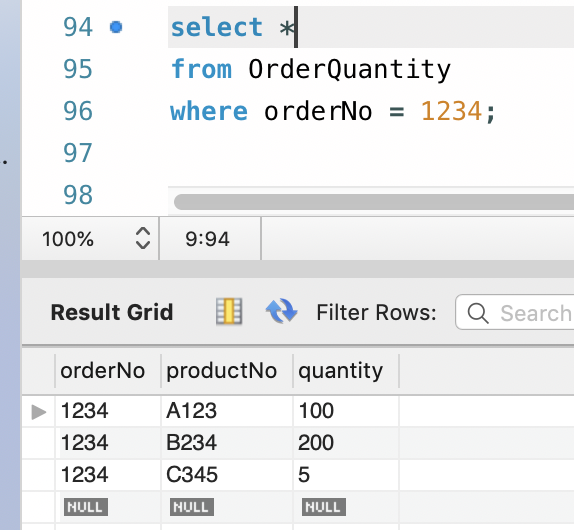
## Insert values into OrderTable table



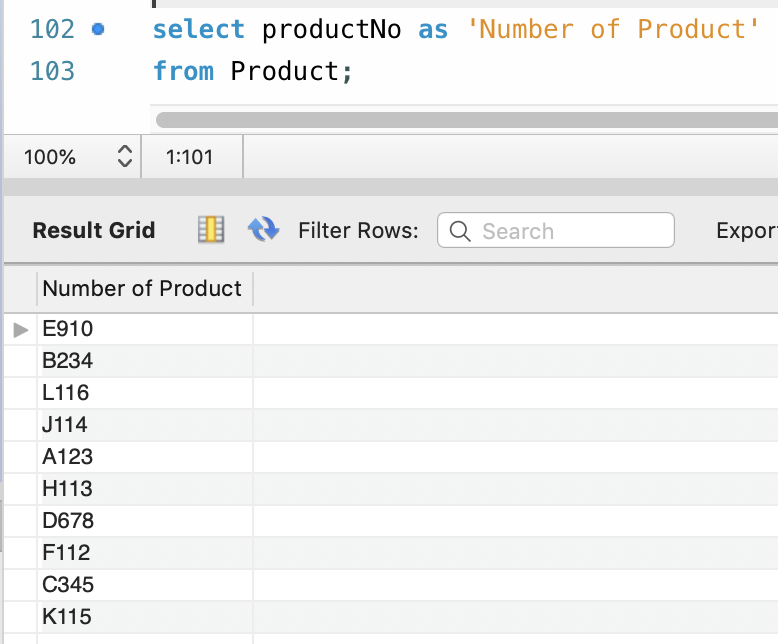
Projection means choosing the attributes of a relation that you want to display and excluding the rest.



Restriction is simply a “filter” that selects rows that meet a specified condition



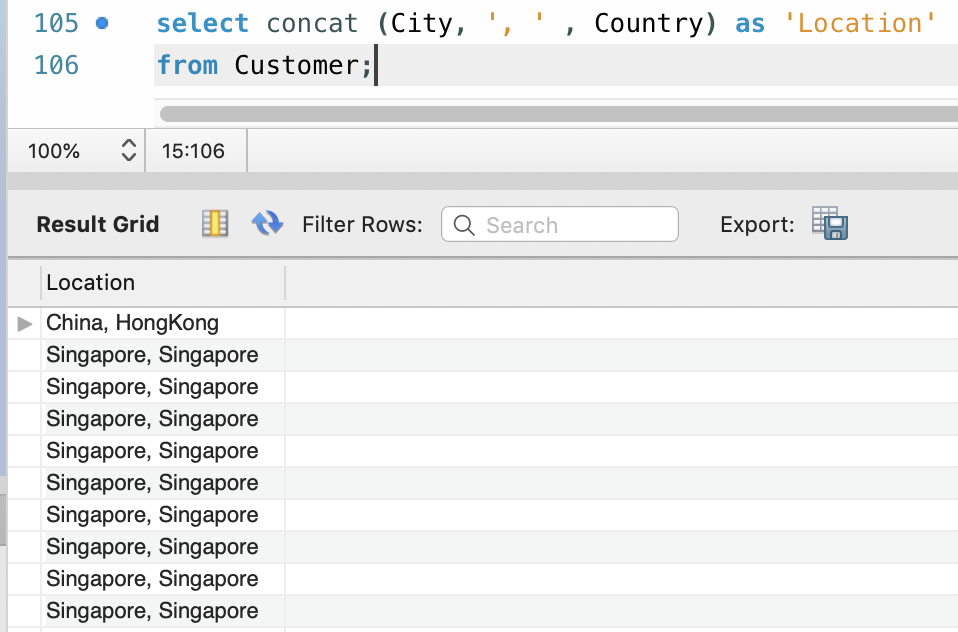
Aliases are used to give a table, or a column in a table, a temporary name.



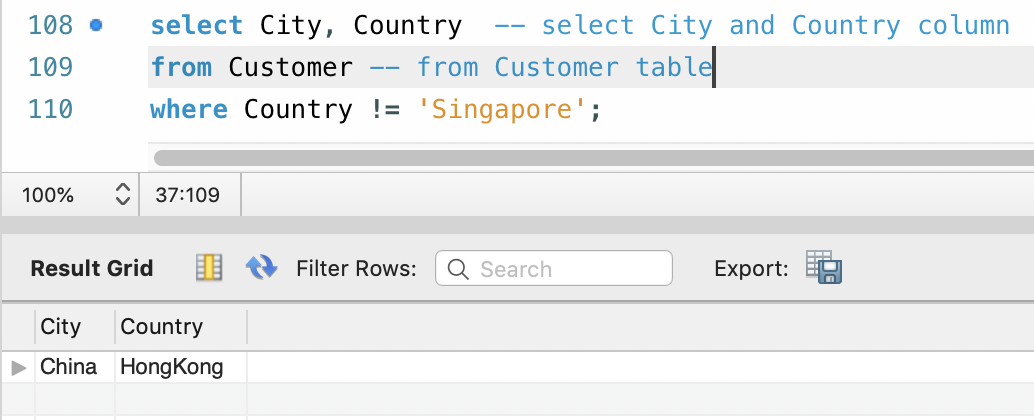
NULL value handling is the term used to find a missing value or check whether a column has a missing value or not.



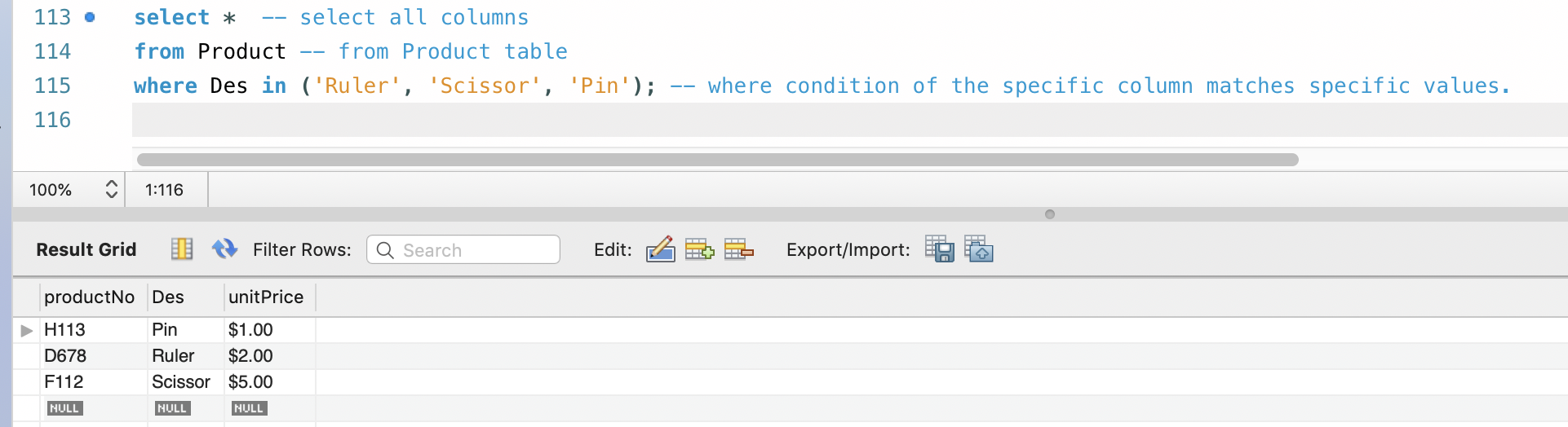
Concatenation provides to concatenate two or more than two character expressions into a single string.



Comparison Operator is a mathematical symbol which is used to compare two values. Comparison operators are used in conditions that compares one expression with another.



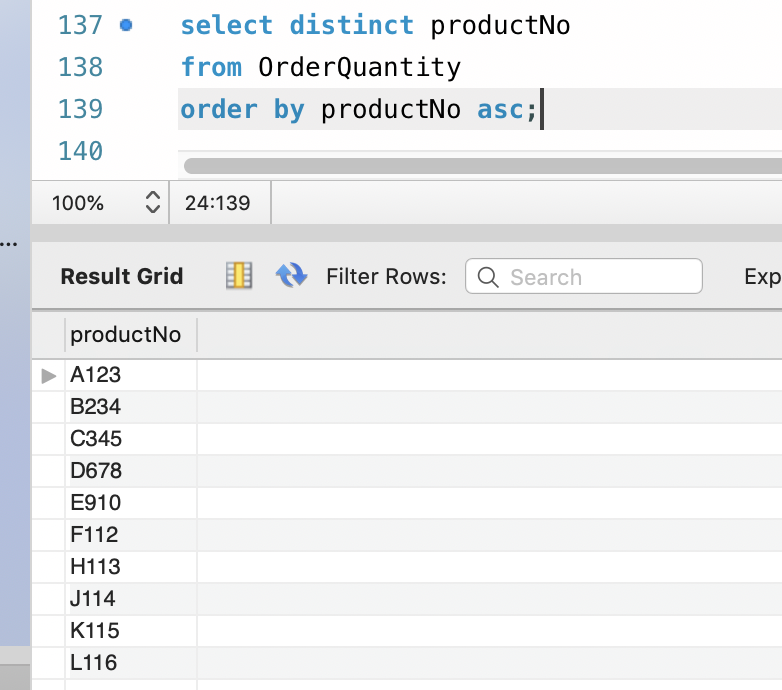
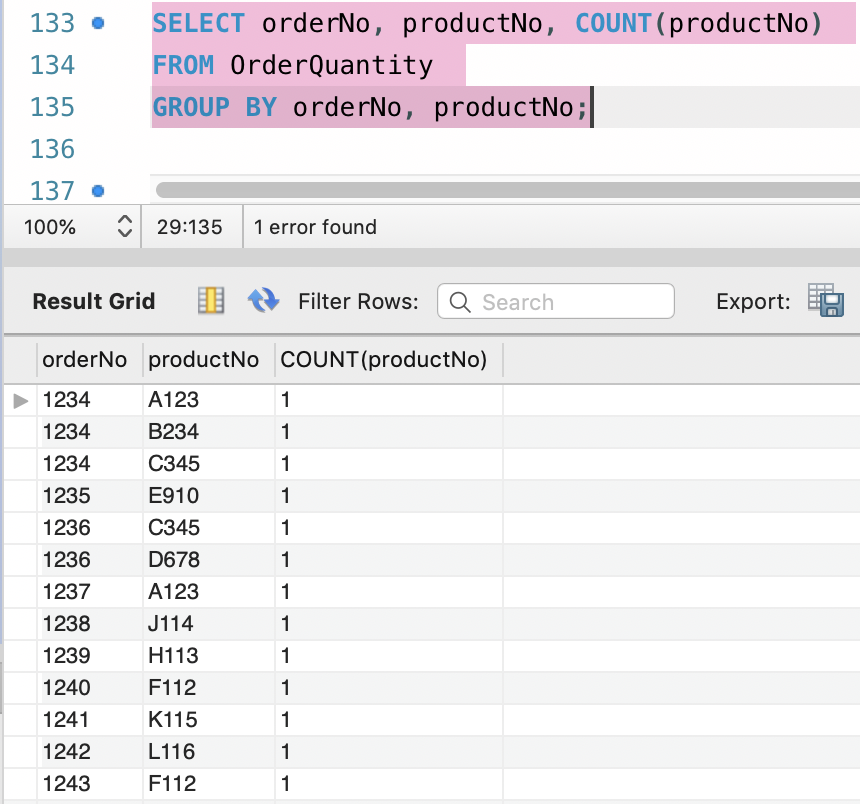
Logical Operator are those that are true or false. They return a true or false values to combine one or more true or false values.

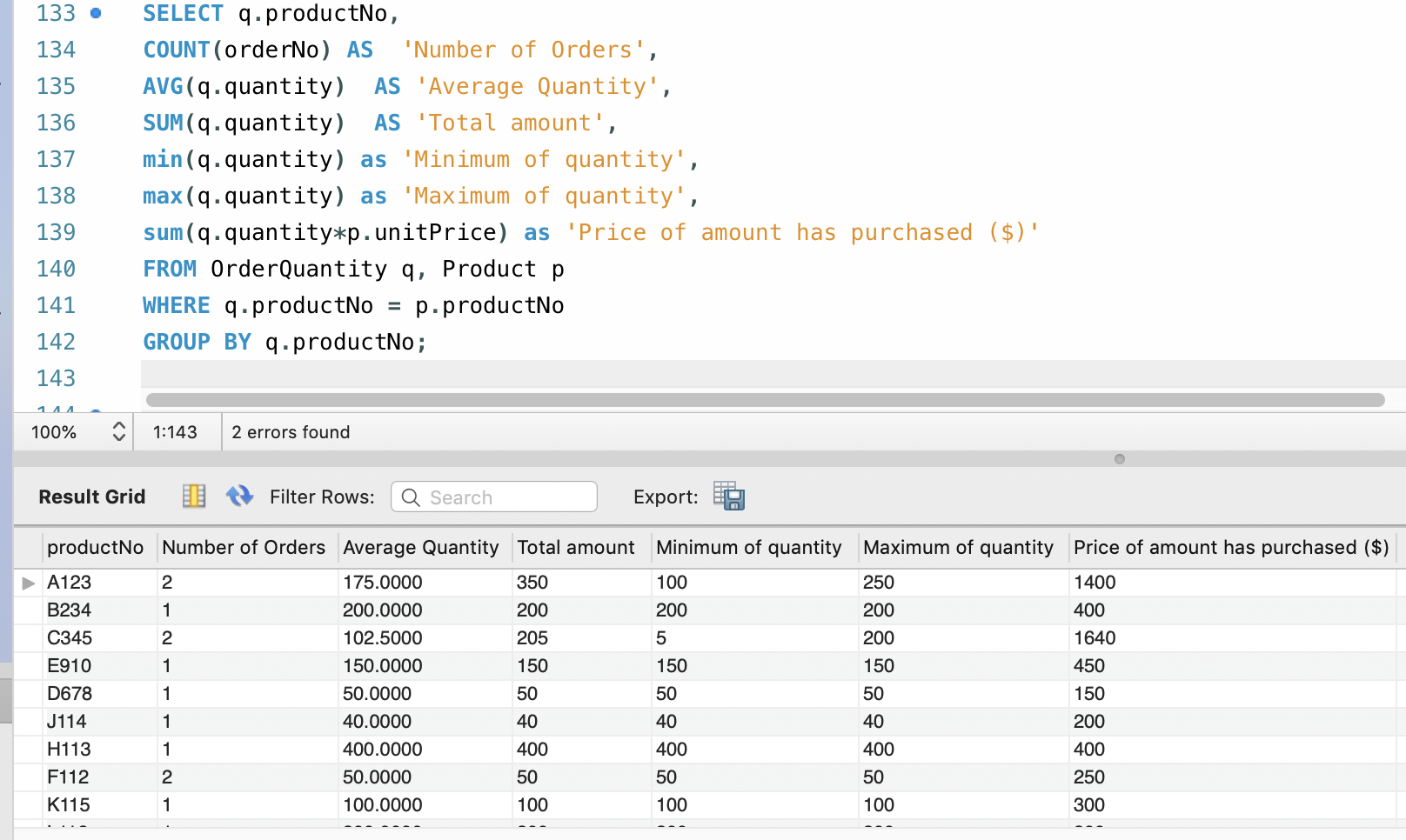


Sorting is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.



Function () Group function & Single Row Function that operate on groups of rows and return one value for the entire group. These functions are: COUNT, MAX, MIN, AVG, SUM, DISTINCT. SQL COUNT (): This function returns the number of rows in the table that satisfies the condition specified in the WHERE condition.



Joins and subqueries are both used to combine data from different tables into a single result. Subqueries can be used to return either a scalar (single) value or a row set; whereas, joins are used to return rows. A common use for a subquery may be to calculate a summary value for use in a query.

