**武汉大学计算机学院**

**本科生实验报告**

**2023春季《数据库系统》课后实验**

专 业 名 称 ：软件工程

课 程 名 称 ：数据库系统

指 导 教 师 ：王黎维

学 生 学 号 ：2021302111204

学 生 姓 名 ：朱俊杰

二○二三年五月

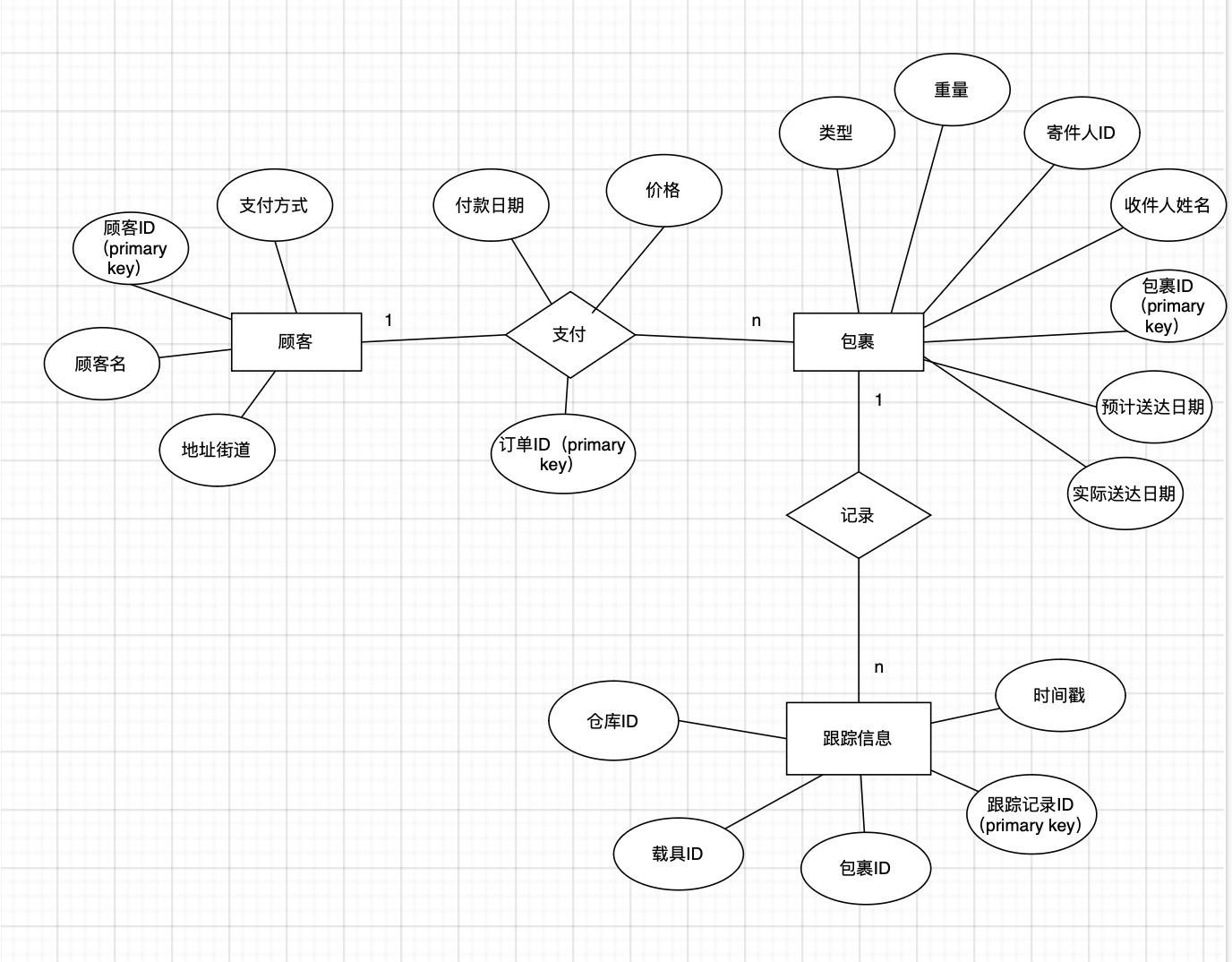
**1实验内容**

本次实验的目标是实践一个小型关系数据库的概念设计、逻辑设计、实现、操作和维护。实验模拟了一个快递公司的数据库应用。

1. 针对包裹的类型（例如平信、小包、大包等）、包裹重量以及投递的时效性（当日达、隔日达等等）可能会有不同种类的服务。
2. 一些客户和快递公司有协议，因此他们的寄递服务费用都会从一个账号中扣款，而且这种扣款是月结。其他的客户都是不那么频繁使用寄递业务的客户，他们会通过信用卡、支付宝、微信等方式支付，并且是一次一结的预付费方式。还有些寄递是为了退换货。
3. 对于大部分的包裹，快递公司并不在意寄送的是什么东西。但是，有些情况是相关法律规定需要注意的：
   1. 危险品，不能走空运
   2. 国际快递，需要客户申报包裹内容以及价值
4. 快递公司需要从包裹被下单开始就进行跟踪，直到包裹被签收。如果有寄给某个人的包裹，他能看到包裹的各个细节：包裹现在在哪里、它从哪里来、它将向哪里去。除了客户能看到的信息，快递公司本身还需要看到更多细节，包括但不限于：包裹当前正在哪辆卡车或者飞机上，或者正在哪个中转仓库。
5. 跟踪功能不仅仅关心当前的事情，例如快递公司可能还需要回看包裹昨天在哪里。还可能会从卡车或者中转仓库的角度来查看包裹，例如查看某个中转仓库当前有哪些包裹、今天有多少包裹从这里发出。
6. 除了包裹跟踪之外，快递公司还有一些其他方面的操作，例如安排卡车和飞机的路线、为卡车和飞机安排驾驶员等等。对于本次实验，我们只考虑包裹处理和付款方面的操作。

**2实验要求与结果实现**

1. **E-R模型**



1. **关系模型**
2. 将E-R图转换成对应的关系模型，运用关系规范化理论知识将它优化为3NF。

Customer（CustomerID , payMethod, name, address）

Receipt (ReceiptID, PaymentDate, amount，CustomerID，PackageID)

Package (PackageID, ExceptedDate，ExactDate, ReceiverName, SenderID, type, weight)

TrackingInformation(TrackingID, PackageID, warehouseID，TrunkID，TrackingTime)

关系规范化理论3NF分析：

由于三个关系的主属性都只有一个，则关系模型一定满足2NF

Customer（CustomerID , payMethod, name, address）非主属性不传递依赖于主属性

Receipt (ReceiptID, PaymentDate, amount，CustomerID，PackageID) 非主属性不传递依赖于主属性

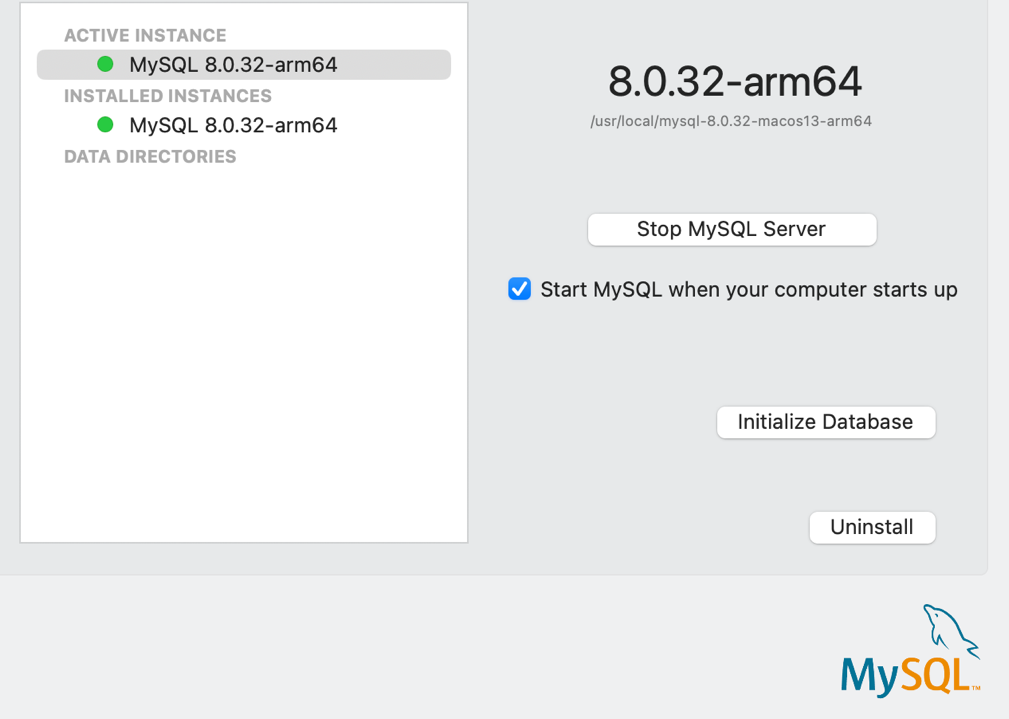
Package (PackageID, ExceptedDate，ExactDate, ReceiverName, SenderID, type, weight) 非主属性不传递依赖于主属性

TrackingInformation(TrackingID, PackageID, warehouseID，TrunkID，TrackingTime) 非主属性不传递依赖于主属性

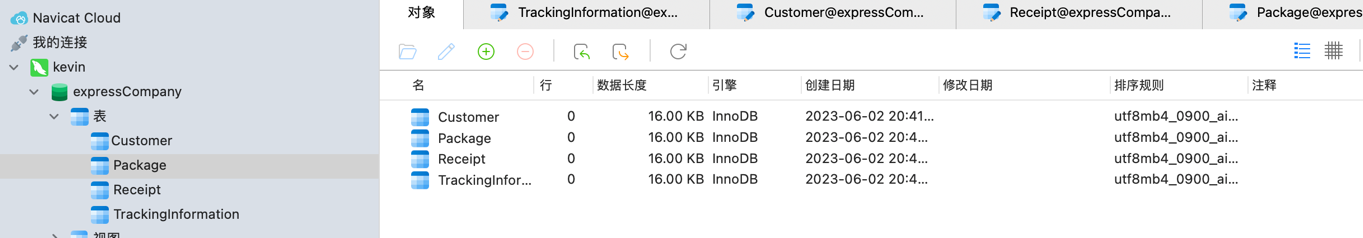
所以关系模型一定满足3NF

1. 将关系模型实现在一中学生熟悉的RDBMS中。

使用RDMBS Mysql 8.0.32-arm64



创建数据库截图展示：



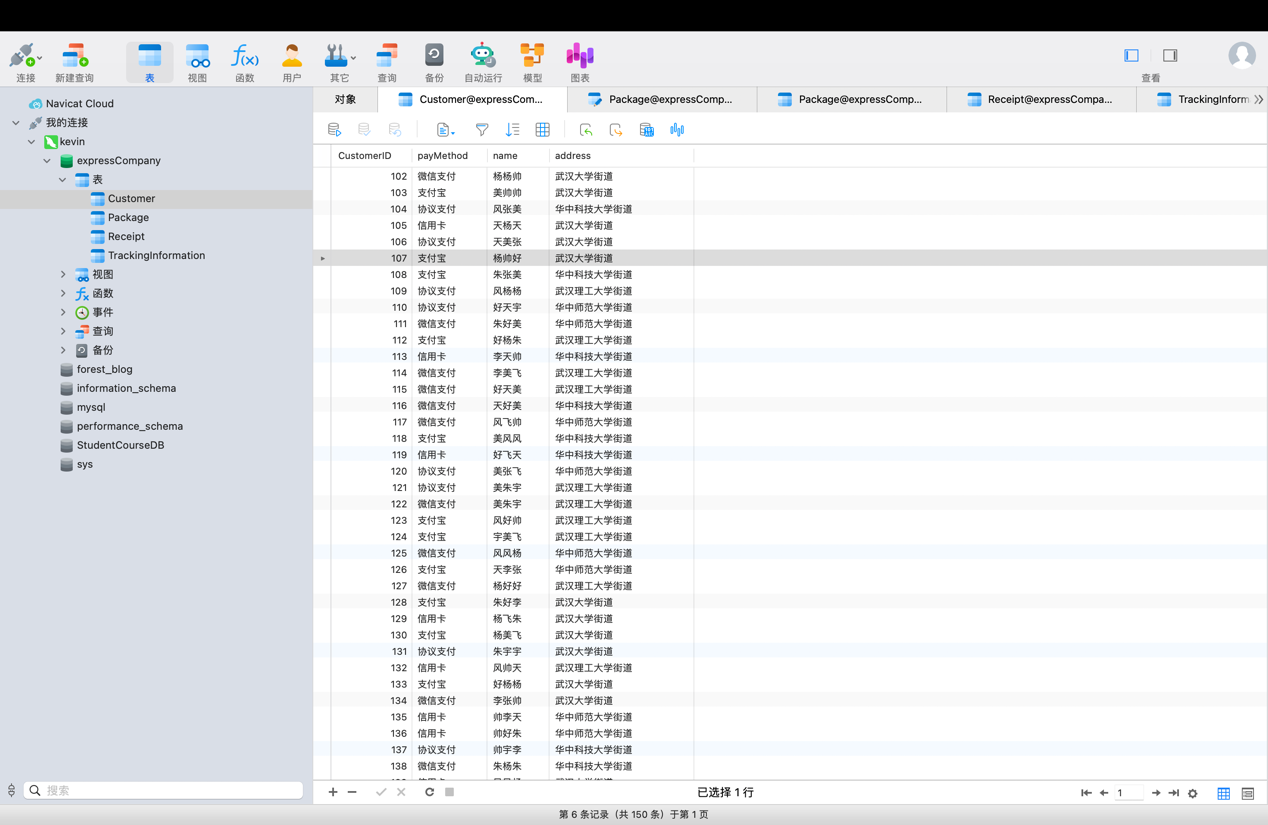
1. 在数据库中创建适当的索引和约束。

|  |
| --- |
| -- 创建Package表约束和索引  ALTER TABLE Package  ADD CONSTRAINT CHK\_Type CHECK (type IN ('危险品', '国际快递', '普通国内快递'));  ALTER TABLE Package  ADD PRIMARY KEY (PackageID);  -- 创建Customer表约束和索引  ALTER TABLE Customer  ADD CONSTRAINT CHK\_PayMethod CHECK (payMethod IN ('协议支付', '支付宝', '信用卡', '微信支付'));  ALTER TABLE Customer  ADD PRIMARY KEY (CustomerID);  -- 创建Receipt表约束和索引  ALTER TABLE Receipt  ADD CONSTRAINT FK\_Receipt\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);  ALTER TABLE Receipt  ADD CONSTRAINT FK\_Receipt\_Package FOREIGN KEY (PackageID) REFERENCES Package(PackageID);  ALTER TABLE Receipt  ADD PRIMARY KEY (ReceiptID);  -- 创建TrackingInformation表约束和索引  ALTER TABLE TrackingInformation  ADD CONSTRAINT FK\_TrackingInformation\_Package FOREIGN KEY (PackageID) REFERENCES Package(PackageID);  ALTER TABLE TrackingInformation  ADD PRIMARY KEY (TrackingID); |

1. **填充关系**

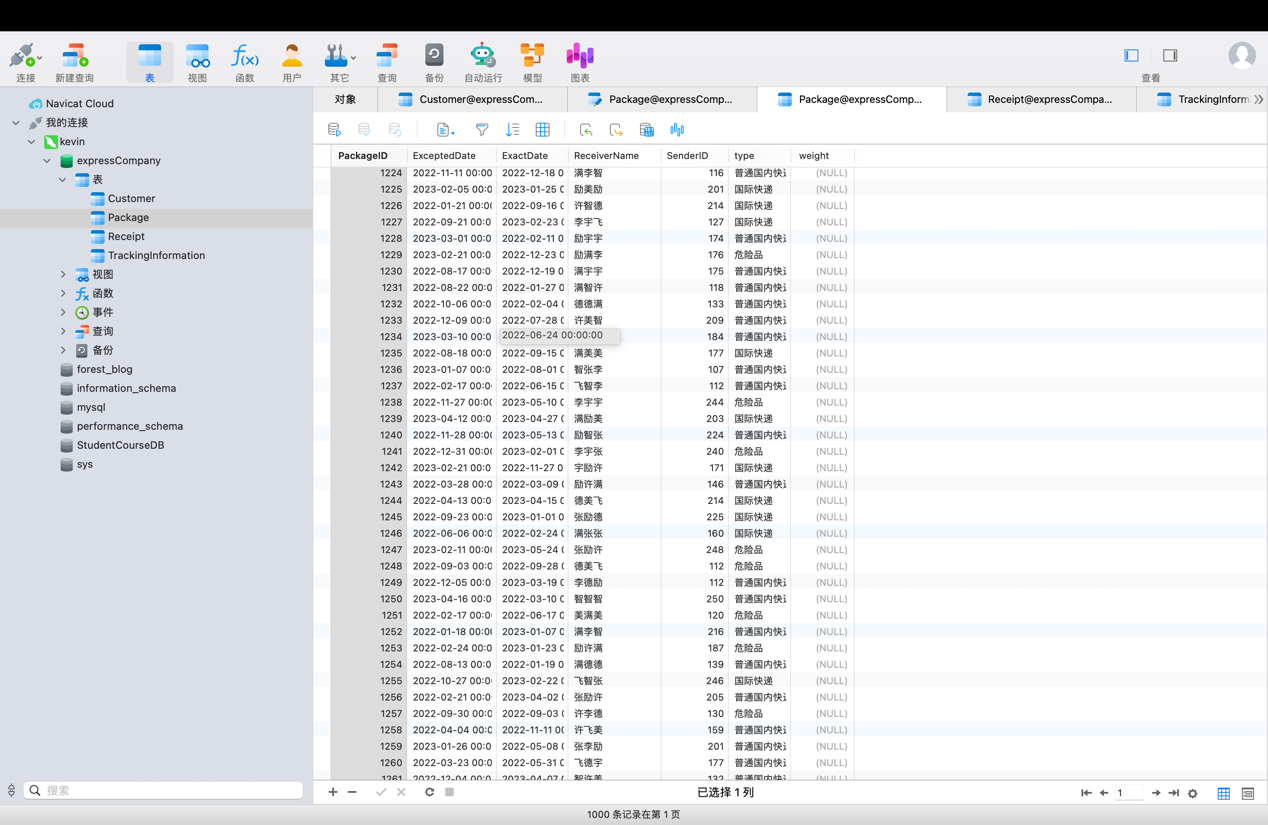
**填充Customer （多次执行插入了150行左右数据）**

|  |
| --- |
| import random import string import mysql.connector # 导入MySQL连接器模块  # 连接数据库 conn = mysql.connector.connect(  host='localhost',  user='root',  password='zjj2003915',  database='expressCompany' )  cursor = conn.cursor()  # 生成随机字符串 def generate\_random\_string(length):  letters = string.ascii\_letters  return ''.join(random.choice(letters) for \_ in range(length))   def generate\_random\_chinese(length):  characters = '朱张李美好天杨帅宇风飞'  return ''.join(random.choice(characters) for \_ in range(length))   # 插入50条数据 for \_ in range(50):  pay\_method = random.choice(['协议支付', '支付宝', '信用卡', '微信支付'])  name = generate\_random\_chinese(3)  address = random.choice(['武汉大学街道', '华中科技大学街道', '华中师范大学街道', '武汉理工大学街道'])   # 执行插入语句  insert\_query = "INSERT INTO Customer (payMethod, name, address) VALUES (%s, %s, %s)"  values = (pay\_method, name, address)  cursor.execute(insert\_query, values)  # 提交事务并关闭连接 conn.commit() conn.close() |
|  |



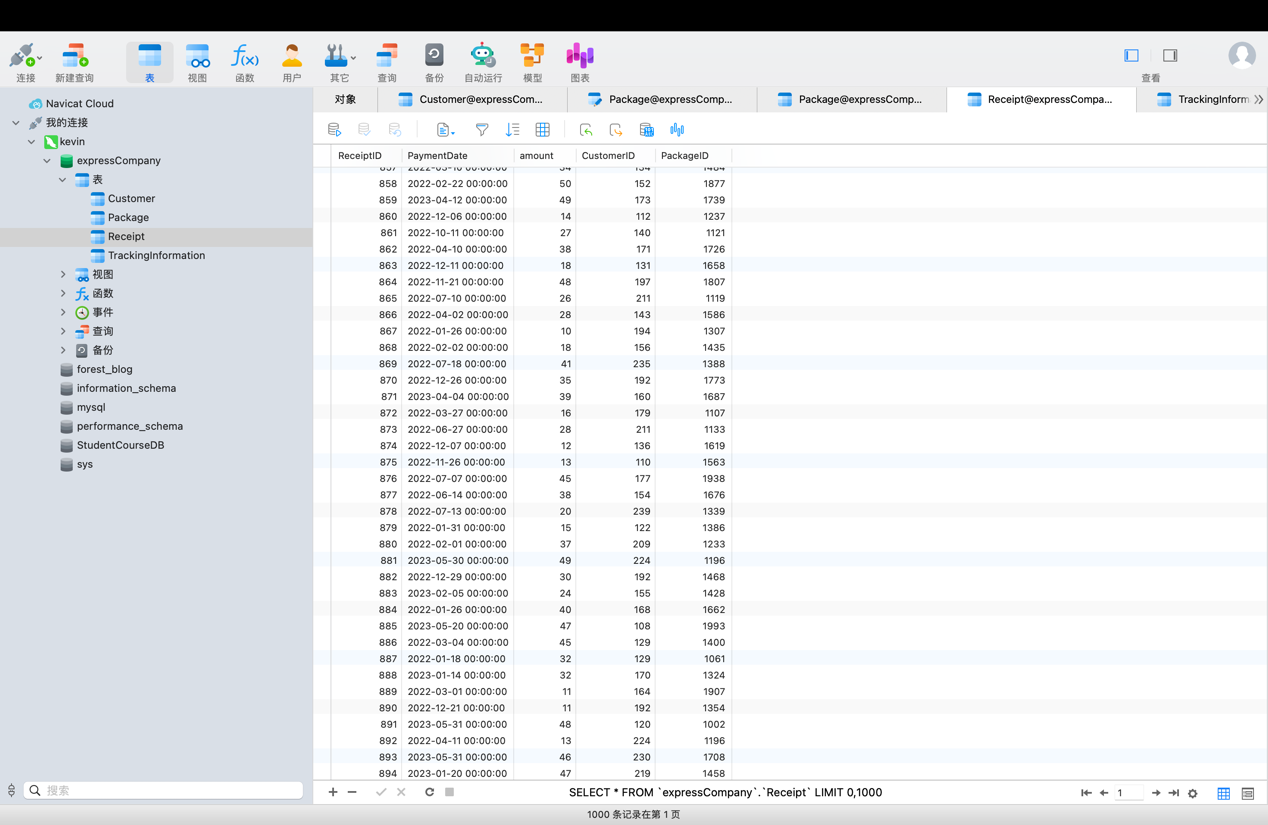
**填充package （1000条记录）**

|  |
| --- |
| import random import datetime import mysql.connector  # 连接数据库 conn = mysql.connector.connect(  host='localhost',  user='root',  password='zjj2003915',  database='expressCompany' )  cursor = conn.cursor()   def generate\_random\_chinese(length):  characters = '张许李满智飞宇美励德'  return ''.join(random.choice(characters) for \_ in range(length))  # 生成随机日期 def generate\_random\_date(start\_date, end\_date):  days = (end\_date - start\_date).days  random\_days = random.randint(0, days)  return start\_date + datetime.timedelta(days=random\_days)  # 插入1000条数据 for \_ in range(1000):  expected\_date = generate\_random\_date(datetime.datetime(2022, 1, 1), datetime.datetime(2023, 5, 31))  exact\_date = generate\_random\_date(datetime.datetime(2022, 1, 1), datetime.datetime(2023, 5, 31))  receiver\_name = generate\_random\_chinese(3)  sender\_id = random.randint(102, 251)  package\_type = random.choice(['危险品', '国际快递', '普通国内快递'])  # 执行插入语句  insert\_query = "INSERT INTO Package (ExceptedDate, ExactDate, ReceiverName, SenderID, type) VALUES (%s, %s, %s, %s, %s)"  values = (expected\_date, exact\_date, receiver\_name, sender\_id, package\_type)  cursor.execute(insert\_query, values)  # 提交事务并关闭连接 conn.commit() conn.close() |



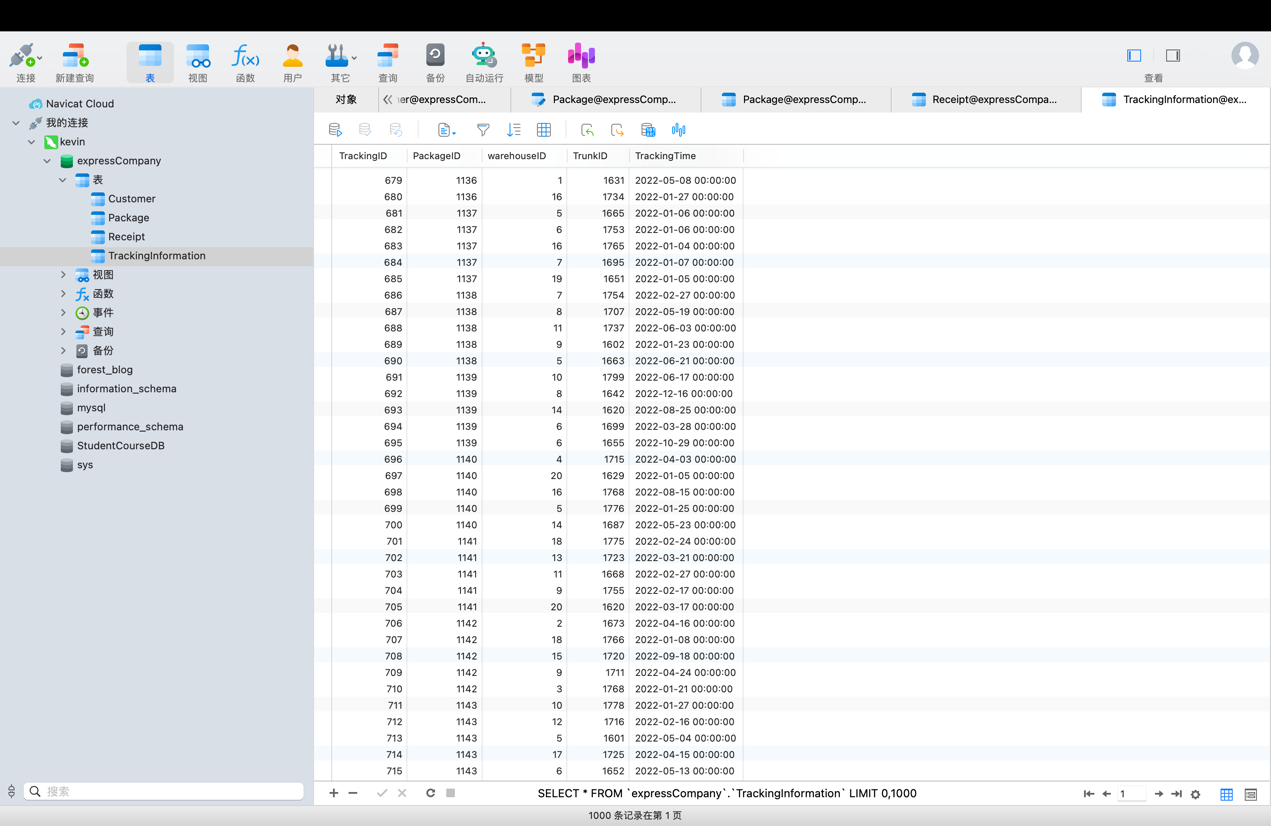
**填充receipt （1000条记录）**

|  |
| --- |
| import random import datetime import mysql.connector  # 连接数据库 conn = mysql.connector.connect(  host='localhost',  user='root',  password='zjj2003915',  database='expressCompany' )  cursor = conn.cursor()   # 生成随机日期 def generate\_random\_date(start\_date, end\_date):  days = (end\_date - start\_date).days  random\_days = random.randint(0, days)  return start\_date + datetime.timedelta(days=random\_days)   # 插入1000条数据 for \_ in range(1000):  payment\_date = generate\_random\_date(datetime.datetime(2022, 1, 1), datetime.datetime(2023, 5, 31))  amount = round(random.uniform(10, 50), 2)   # 从Package表中随机选择一条记录  cursor.execute("SELECT SenderID, PackageID FROM Package ORDER BY RAND() LIMIT 1")  result = cursor.fetchone()  sender\_id = result[0]  package\_id = result[1]   # 执行插入语句  insert\_query = "INSERT INTO Receipt (PaymentDate, amount, CustomerID, PackageID) VALUES (%s, %s, %s, %s)"  values = (payment\_date, amount, sender\_id, package\_id)  cursor.execute(insert\_query, values)  # 提交事务并关闭连接 conn.commit() conn.close() |



**填充TrackingInformation（5000条记录）**

|  |
| --- |
| import random import datetime import mysql.connector  # 连接数据库 conn = mysql.connector.connect(  host='localhost',  user='root',  password='zjj2003915',  database='expressCompany' )  cursor = conn.cursor()   # 生成随机日期 def generate\_random\_date(start\_date, end\_date):  days = (end\_date - start\_date).days  random\_days = random.randint(0, days)  return start\_date + datetime.timedelta(days=random\_days)   # 从Package表中获取所有记录按顺序 cursor.execute("SELECT PackageID, ExactDate FROM Package ORDER BY PackageID") packages = cursor.fetchall()  # 插入5000条数据 for package in packages:  package\_id = package[0]  exact\_date = package[1]   # 插入5条数据  for \_ in range(5):  warehouse\_id = random.randint(1, 20)  trunk\_id = random.randint(1600, 1800)  tracking\_time = generate\_random\_date(datetime.datetime(2022, 1, 1), exact\_date)   # 执行插入语句  insert\_query = "INSERT INTO TrackingInformation (PackageID, warehouseID, TrunkID, TrackingTime) VALUES (%s, %s, %s, %s)"  values = (package\_id, warehouse\_id, trunk\_id, tracking\_time)  cursor.execute(insert\_query, values)  # 提交事务并关闭连接 conn.commit() conn.close() |

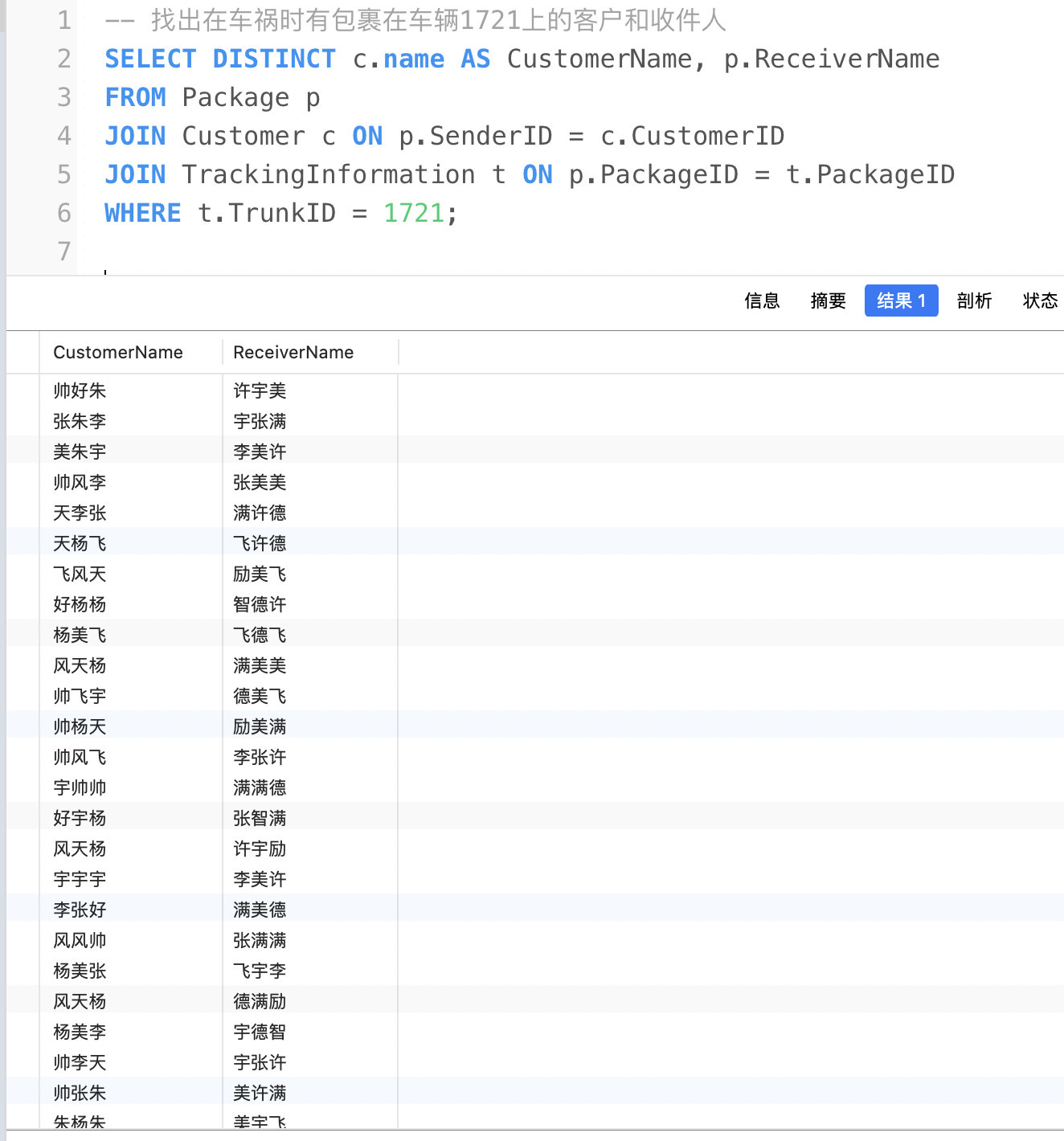


**4) 查询**

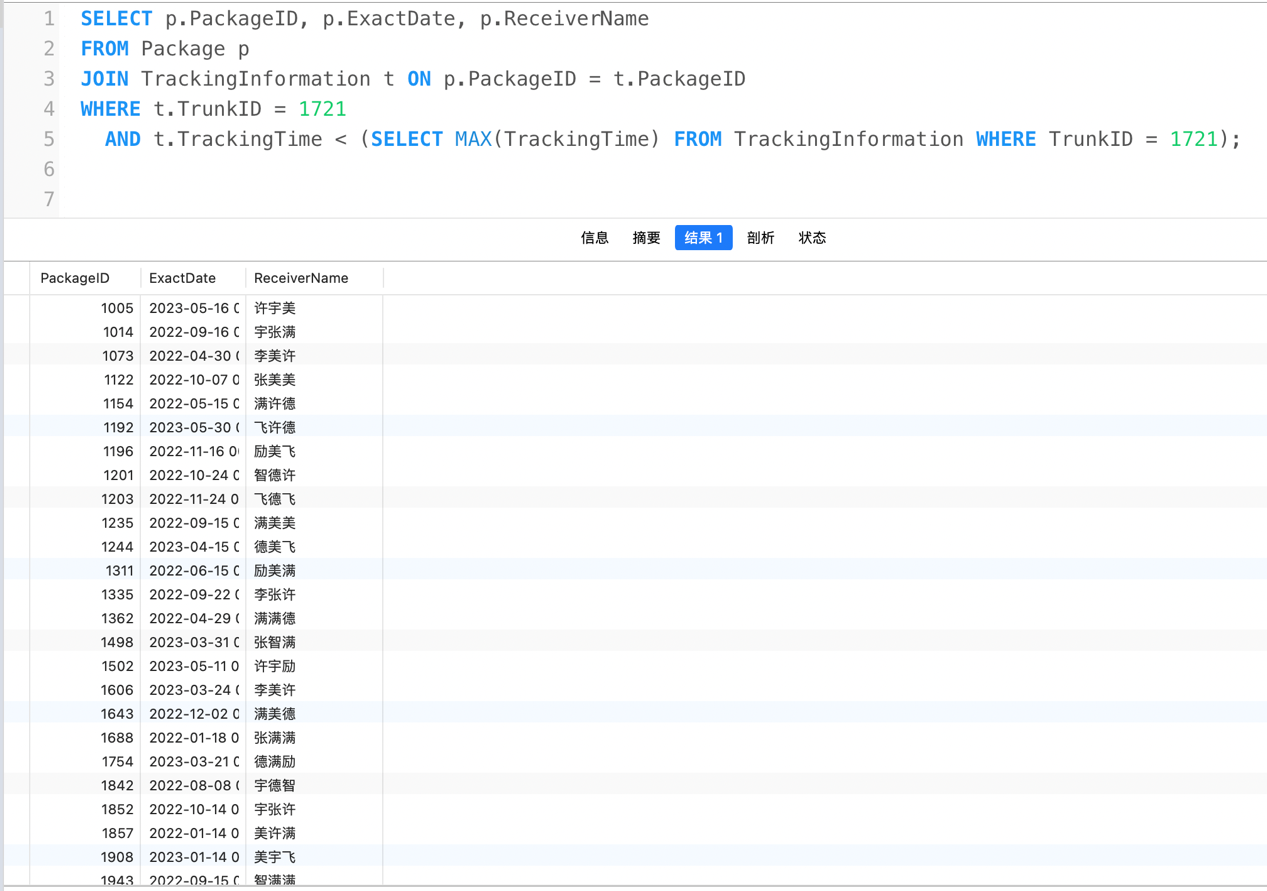
学生需要运行一些测试查询来展示其数据库中的数据。下面是查询的需求：

**a) 假设卡车1721在车祸中被烧毁了。找出所有在车祸时有包裹在该车上的客户和收件人。找出该车在车祸前最后一次成功投递包裹的情况。**

|  |
| --- |
| -- 找出在车祸时有包裹在车辆1721上的客户和收件人  SELECT DISTINCT c.name AS CustomerName, p.ReceiverName  FROM Package p  JOIN Customer c ON p.SenderID = c.CustomerID  JOIN TrackingInformation t ON p.PackageID = t.PackageID  WHERE t.TrunkID = 1721; |

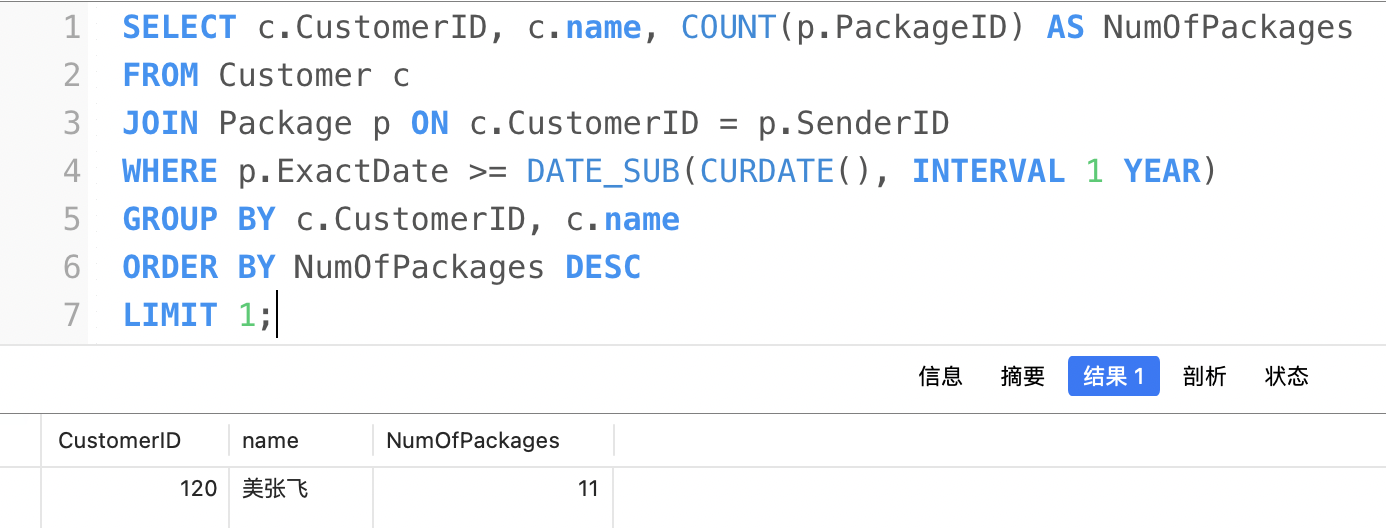


|  |
| --- |
| 找出该车在车祸前最后一次成功投递包裹的情况  SELECT p.PackageID, p.ExactDate, p.ReceiverName  FROM Package p  JOIN TrackingInformation t ON p.PackageID = t.PackageID  WHERE t.TrunkID = 1721  AND t.TrackingTime < (SELECT MAX(TrackingTime) FROM TrackingInformation WHERE TrunkID = 1721); |



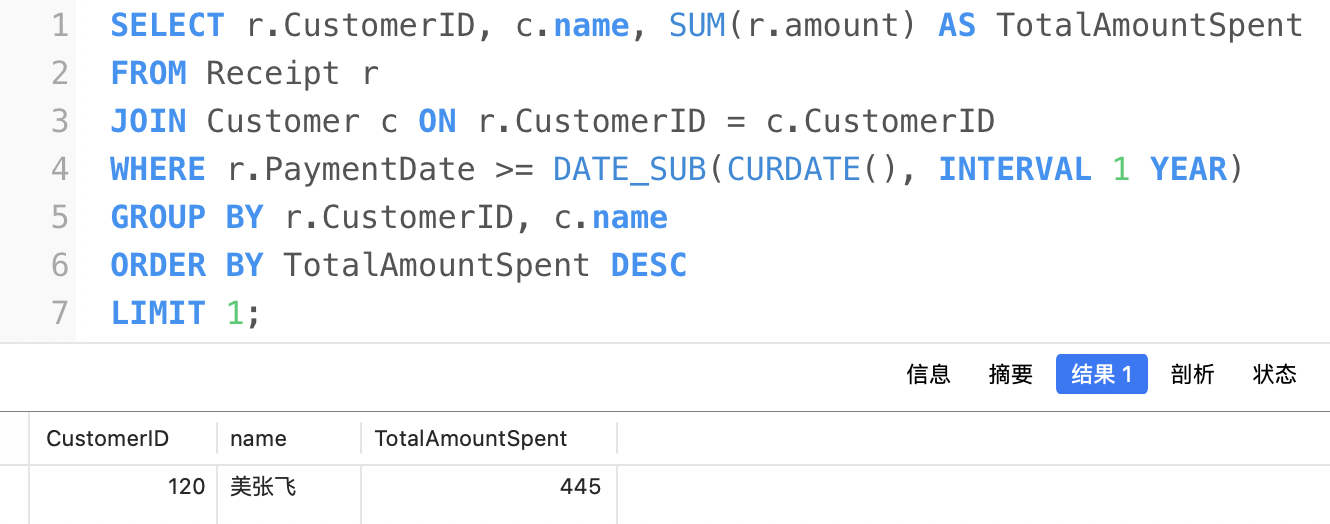
**b) 找出在过去一年中寄出了最多包裹的客户。**

|  |
| --- |
| SELECT c.CustomerID, c.name, COUNT(p.PackageID) AS NumOfPackages  FROM Customer c  JOIN Package p ON c.CustomerID = p.SenderID  WHERE p.ExactDate >= DATE\_SUB(CURDATE(), INTERVAL 1 YEAR)  GROUP BY c.CustomerID, c.name  ORDER BY NumOfPackages DESC  LIMIT 1; |



**c) 找出在过去一年中在寄包裹上花了最多前的客户。**

|  |
| --- |
| SELECT r.CustomerID, c.name, SUM(r.amount) AS TotalAmountSpent  FROM Receipt r  JOIN Customer c ON r.CustomerID = c.CustomerID  WHERE r.PaymentDate >= DATE\_SUB(CURDATE(), INTERVAL 1 YEAR)  GROUP BY r.CustomerID, c.name  ORDER BY TotalAmountSpent DESC  LIMIT 1; |



**d) 找出客户最密集的街道，即找出拥有最多客户的街道。**

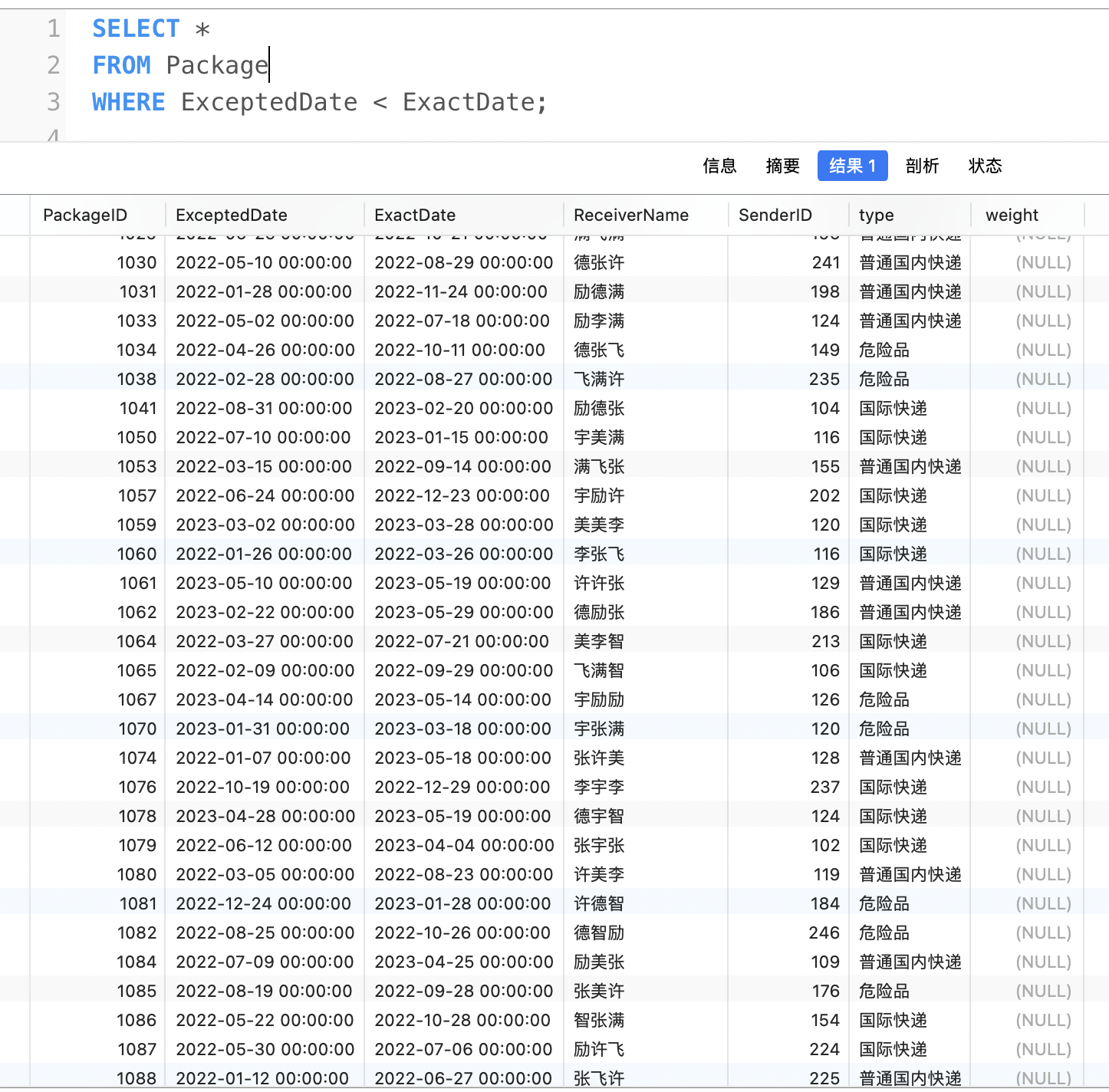
|  |
| --- |
| SELECT address AS Street, COUNT(CustomerID) AS NumOfCustomers  FROM Customer  GROUP BY address  ORDER BY NumOfCustomers DESC  LIMIT 1; |



**e) 找出没有在承诺时间内送达的包裹。**

|  |
| --- |
| SELECT \*  FROM Package  WHERE ExceptedDate < ExactDate; |

⚠️ 这里因为编写脚本时逻辑问题，导致几乎所有都是实际到达时间晚于预期，即没有在承诺时间内送达的包裹，产生的数据如下，大部分迟到



**f) 为每个客户生成一份上月的账单。考虑创建几种类型的账单：**

** 简单账单：客户、地址和应付金额。**

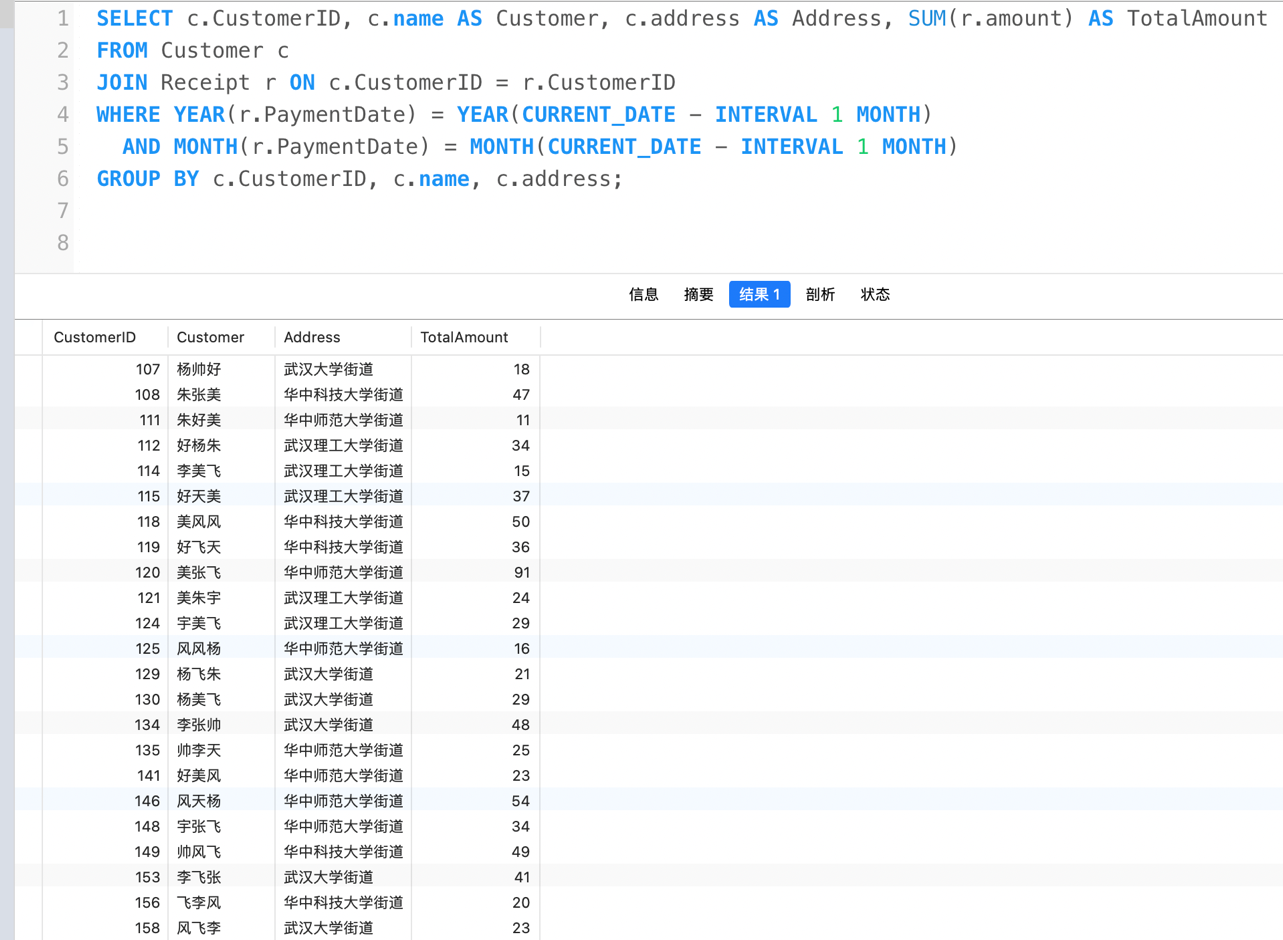
** 列有各类服务收费的账单。**

** 详细账单：列出每一笔单独的寄递和收费。**

**快递公司希望从数据库查询得到的结果在显示格式上尽可能就像一张“账单”。**

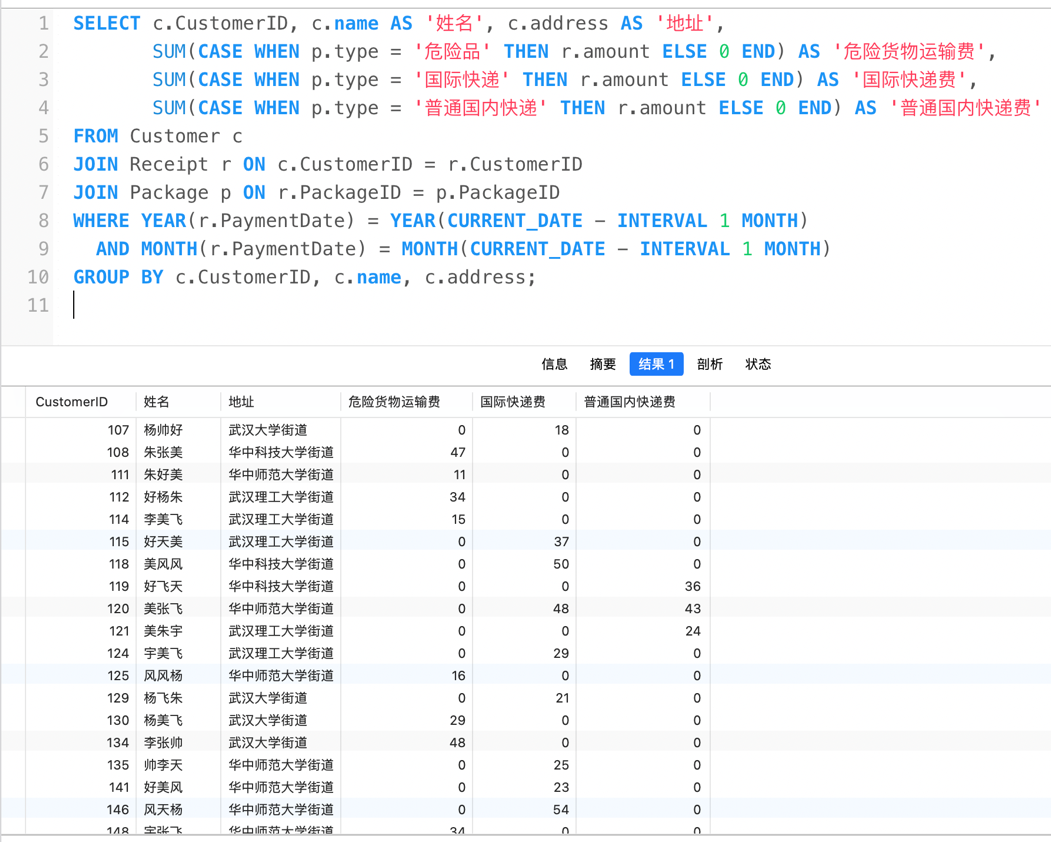
简单账单：

|  |
| --- |
| SELECT c.CustomerID, c.name AS Customer, c.address AS Address, SUM(r.amount) AS TotalAmount  FROM Customer c  JOIN Receipt r ON c.CustomerID = r.CustomerID  WHERE YEAR(r.PaymentDate) = YEAR(CURRENT\_DATE - INTERVAL 1 MONTH)  AND MONTH(r.PaymentDate) = MONTH(CURRENT\_DATE - INTERVAL 1 MONTH)  GROUP BY c.CustomerID, c.name, c.address; |



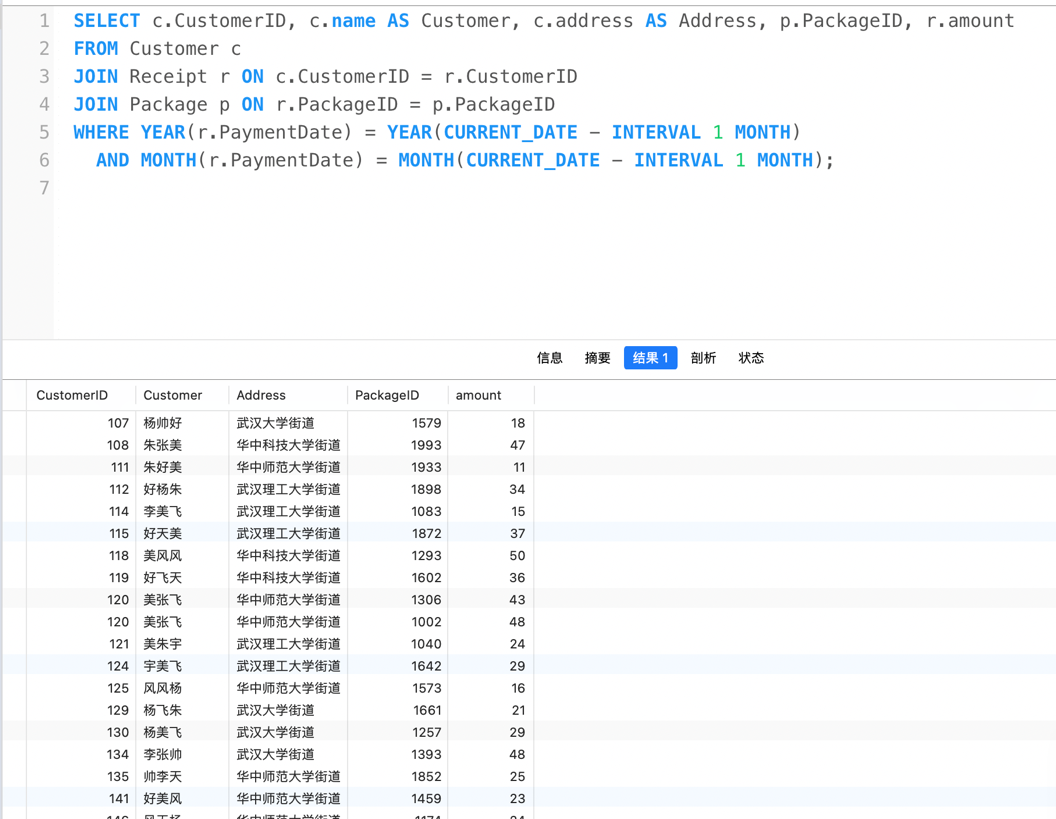
列有各类服务收费的账单：

|  |
| --- |
| SELECT c.CustomerID, c.name AS '姓名', c.address AS '地址',  SUM(CASE WHEN p.type = '危险品' THEN r.amount ELSE 0 END) AS '危险货物运输费',  SUM(CASE WHEN p.type = '国际快递' THEN r.amount ELSE 0 END) AS '国际快递费',  SUM(CASE WHEN p.type = '普通国内快递' THEN r.amount ELSE 0 END) AS '普通国内快递费'  FROM Customer c  JOIN Receipt r ON c.CustomerID = r.CustomerID  JOIN Package p ON r.PackageID = p.PackageID  WHERE YEAR(r.PaymentDate) = YEAR(CURRENT\_DATE - INTERVAL 1 MONTH)  AND MONTH(r.PaymentDate) = MONTH(CURRENT\_DATE - INTERVAL 1 MONTH)  GROUP BY c.CustomerID, c.name, c.address; |



详细账单：

|  |
| --- |
| SELECT c.CustomerID, c.name AS Customer, c.address AS Address, p.PackageID, r.amount  FROM Customer c  JOIN Receipt r ON c.CustomerID = r.CustomerID  JOIN Package p ON r.PackageID = p.PackageID  WHERE YEAR(r.PaymentDate) = YEAR(CURRENT\_DATE - INTERVAL 1 MONTH)  AND MONTH(r.PaymentDate) = MONTH(CURRENT\_DATE - INTERVAL 1 MONTH); |

****

教师评语评分

评语：

评分：

评阅人：

年 月 日