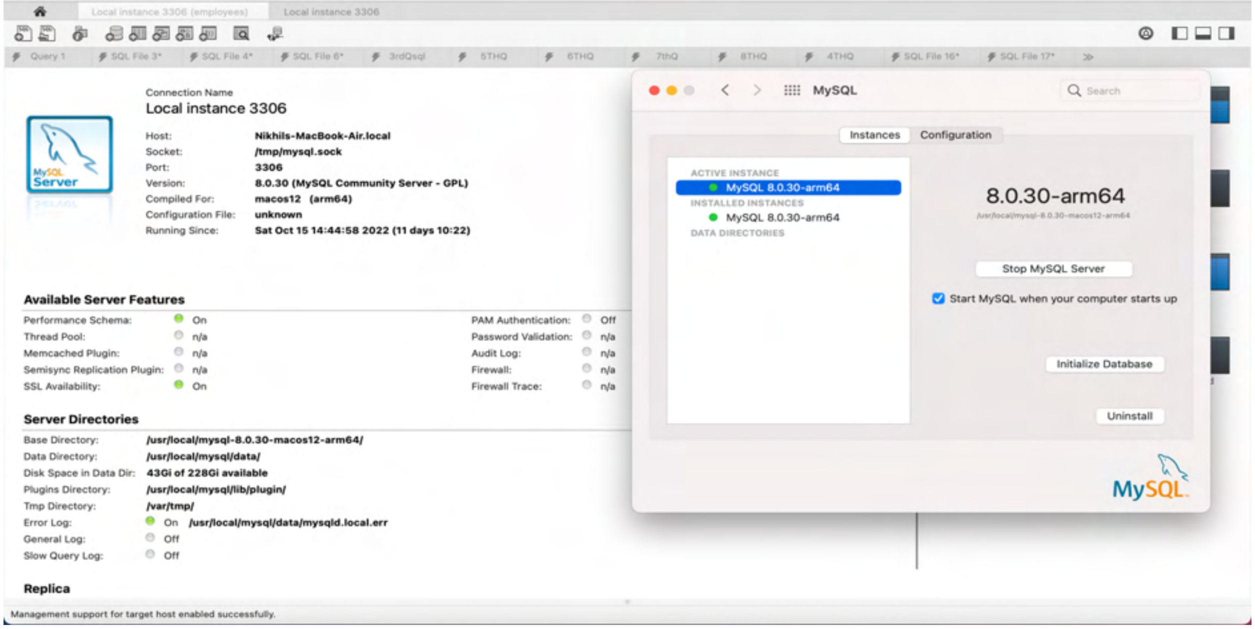
Nikhil Mylarusetty- 016656393

1. [35 points] Python with MySQL
   a. [3 points]

Go through the tutorial at https://realpython.com/python-mysql/.Install MySQL Server (if not already done) and MySQL Connector/Python.

- Installed MYSQL Server and MYSQL Connector successfully.



Connection Name
Local instance 3306

Host: Nikhils-MacBook-Air.local
Socket: /tmp/mysql.sock
Port: 3306
Version: 8.0.30 (MySQL Community Server - GPL)
Compiled For: macos12 (arm64)
Configuration File: unknown
Running Since: Sat Oct 15 14:44:58 2022 (11 days 10:22)

Available Server Features

Performance Schema: ● On
Thread Pool: ○ n/a
Memcached Plugin: ○ n/a
Semisync Replication Plugin: ○ n/a
SSL Availability: ● On
PAM Authentication: ○ Off
Password Validation: ○ n/a
Audit Log: ○ n/a
Firewall: ○ n/a
Firewall Trace: ○ n/a

Server Directories

Base Directory: /usr/local/mysql-8.0.30-macos12-arm64/
Data Directory: /usr/local/mysql/data/
Disk Space in Data Dir: 43Gi of 228Gi available
Plugins Directory: /usr/local/mysql/lib/plugin/
Tmp Directory: /var/tmp/
Error Log: ● On /usr/local/mysql/data/mysqld.local.err
General Log: ○ Off
Slow Query Log: ○ Off

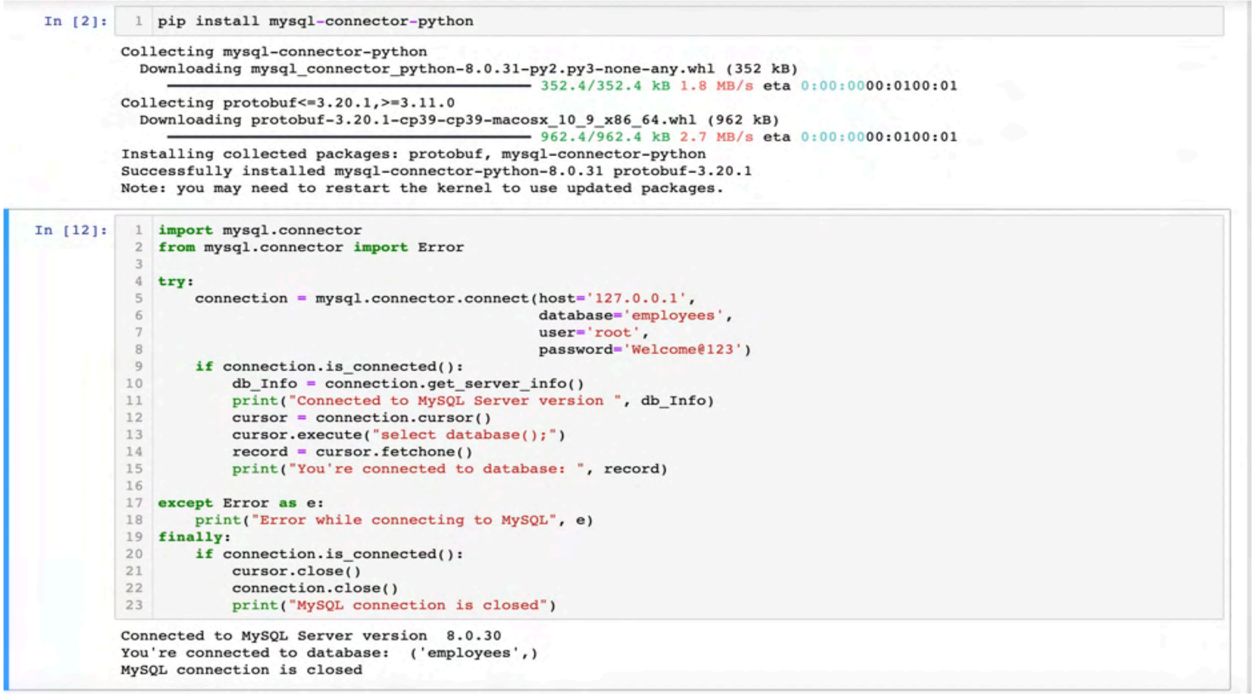
Replica

Management support for target host enabled successfully.

ACTIVE INSTANCE
● MySQL 8.0.30-arm64
INSTALLED INSTANCES
● MySQL 8.0.30-arm64
DATA DIRECTORIES

8.0.30-arm64
/usr/local/mysql-8.0.30-macos12-arm64

☑ Start MySQL when your computer starts up



```
In [2]: pip install mysql-connector-python

Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.31-py2.py3-none-any.whl (352 kB)
                                    ━━━━━━━━━━━━━━━━━━━━ 352.4/352.4 kB 1.8 MB/s eta 0:00:0000:0100:01
Collecting protobuf<=3.20.1,>=3.11.0
  Downloading protobuf-3.20.1-cp39-cp39-macosx_10_9_x86_64.whl (962 kB)
                                    ━━━━━━━━━━━━━━━━━━━━ 962.4/962.4 kB 2.7 MB/s eta 0:00:0000:0100:01
Installing collected packages: protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.31 protobuf-3.20.1
Note: you may need to restart the kernel to use updated packages.
```

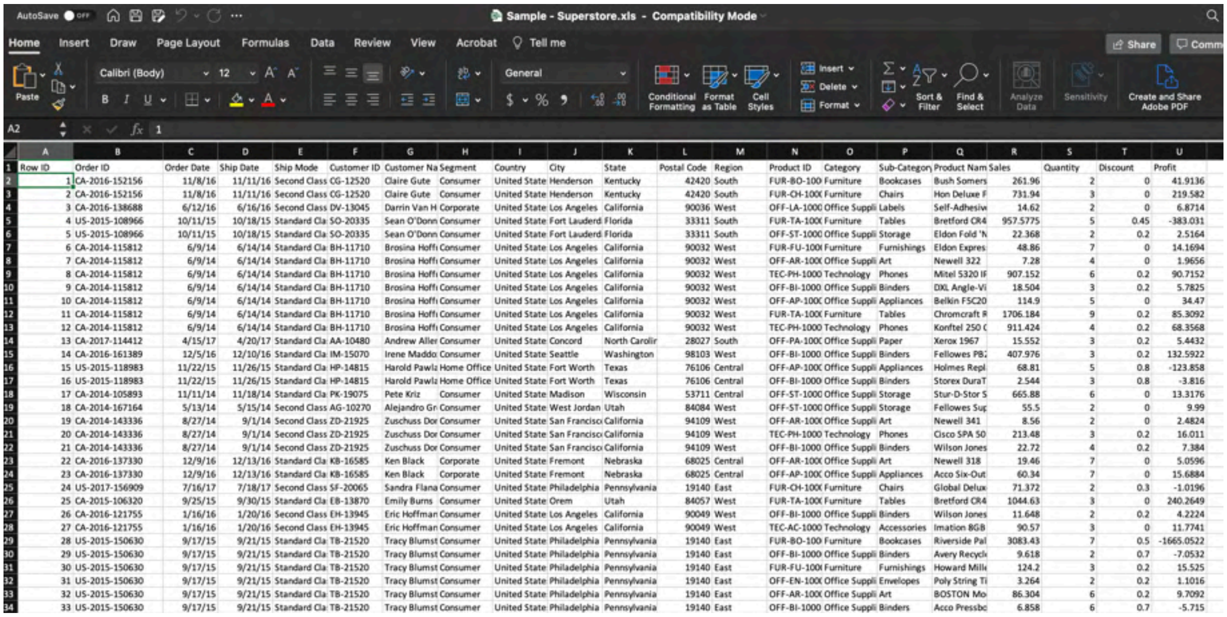
```
In [12]:
import mysql.connector
from mysql.connector import Error

try:
    connection = mysql.connector.connect(host='127.0.0.1',
                                         database='employees',
                                         user='root',
                                         password='Welcome@123')
    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record)

except Error as e:
    print("Error while connecting to MySQL", e)
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

```
Connected to MySQL Server version  8.0.30
You're connected to database:  ('employees',)
MySQL connection is closed
```

Nikhil Mylarusetty- 016656393

b. [7 points]
Choose a dataset, which is not normalized to 3NF such as the one at https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized#.
Create two or more tables (and csv files) from it, so that the tables, when imported into MySQL will be in 3NF. The tables need not be exhaustive – you can leave out columns from the dataset. Justify that the tables are in 3NF.
Superstore sales Dataset: https://www.kaggle.com/datasets/abiodunonadeji/united-state-superstore-sales

- By making sure that there are no transitive dependency relationships among the tables, I converted an unnormalized dataset of retail store details into a third normal form. Below are the DDL statements I used to create the schemas.
- The dataset contains numerous features, focused on the important ones and the rest are not considered.



Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Na	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Categor	Product Nam	Sales	Quantity	Discount	Profit
1	CA-2016-152156	11/8/16	11/11/16	Second Class	CG-12520	Claire Gute	Consumer	United State	Henderson	Kentucky	42420	South	FUR-BO-100	Furniture	Bookcases	Bush Somers	261.96	2	0	41.9136
2	CA-2016-152156	11/8/16	11/11/16	Second Class	CG-12520	Claire Gute	Consumer	United State	Henderson	Kentucky	42420	South	FUR-CH-100(	Furniture	Chairs	Hon Deluxe F	731.94	3	0	219.582
3	CA-2016-138688	6/12/16	6/16/16	Second Class	DV-13045	Darrin Van H	Corporate	United State	Los Angeles	California	90036	West	OFF-LA-1000	Office Suppli	Labels	Self-Adhesiv	14.62	2	0	6.8714
4	US-2015-108966	10/11/15	10/18/15	Standard Cla	SO-20335	Sean O'Donn	Consumer	United State	Fort Lauderd	Florida	33311	South	FUR-TA-100(	Furniture	Tables	Bretford CR4	957.5775	5	0.45	-383.031
5	US-2015-108966	10/11/15	10/18/15	Standard Cla	SO-20335	Sean O'Donn	Consumer	United State	Fort Lauderd	Florida	33311	South	OFF-ST-1000	Office Suppli	Storage	Eldon Fold 'N	22.368	2	0.2	2.5164
6	CA-2014-115812	6/9/14	6/14/14	Standard Cla	BH-11710	Brosina Hoffi	Consumer	United State	Los Angeles	California	90032	West	FUR-FU-100(	Furniture	Furnishings	Eldon Expres	48.86	7	0	14.1694
7	CA-2014-115812	6/9/14	6/14/14	Standard Cla	BH-11710	Brosina Hoffi	Consumer	United State	Los Angeles	California	90032	West	OFF-AR-100(	Office Suppli	Art	Newell 322	7.28	4	0	1.9656
8	CA-2014-115812	6/9/14	6/14/14	Standard Cla	BH-11710	Brosina Hoffi	Consumer	United State	Los Angeles	California	90032	West	TEC-PH-1000	Technology	Phones	Mitel 5320 IF	907.152	6	0.2	90.7152
9	CA-2014-115812	6/9/14	6/14/14	Standard Cla	BH-11710	Brosina Hoffi	Consumer	United State	Los Angeles	California	90032	West	OFF-BI-1000	Office Suppli	Binders	DXL Angle-Vi	18.504	3	0.2	5.7825
10	CA-2014-115812	6/9/14	6/14/14	Standard Cla	BH-11710	Brosina Hoffi	Consumer	United State	Los Angeles	California	90032	West	OFF-AP-100(	Office Suppli	Appliances	Belkin F5C20	114.9	5	0	34.47
11	CA-2014-115812	6/9/14	6/14/14	Standard Cla	BH-11710	Brosina Hoffi	Consumer	United State	Los Angeles	California	90032	West	FUR-TA-100(	Furniture	Tables	Chromcraft R	1706.184	9	0.2	85.3092
12	CA-2014-115812	6/9/14	6/14/14	Standard Cla	BH-11710	Brosina Hoffi	Consumer	United State	Los Angeles	California	90032	West	TEC-PH-1000	Technology	Phones	Konftel 250 (	911.424	4	0.2	68.3568
13	CA-2017-114412	4/15/17	4/20/17	Standard Cla	AA-10480	Andrew Aller	Consumer	United State	Concord	North Carolir	28027	South	OFF-PA-100(	Office Suppli	Paper	Xerox 1967	15.552	3	0.2	5.4432
14	CA-2016-161389	12/5/16	12/10/16	Standard Cla	IM-15070	Irene Maddo:	Consumer	United State	Seattle	Washington	98103	West	OFF-BI-1000	Office Suppli	Binders	Fellowes PB2	407.976	3	0.2	132.5922
15	US-2015-118983	11/22/15	11/26/15	Standard Cla	HP-14815	Harold Pawlz	Home Office	United State	Fort Worth	Texas	76106	Central	OFF-AP-100(	Office Suppli	Appliances	Holmes Repl	68.81	5	0.8	-123.858
16	US-2015-118983	11/22/15	11/26/15	Standard Cla	HP-14815	Harold Pawlz	Home Office	United State	Fort Worth	Texas	76106	Central	OFF-BI-1000	Office Suppli	Binders	Storex DuraT	2.544	3	0.8	-3.816
17	CA-2014-105893	11/11/14	11/18/14	Standard Cla	PK-19075	Pete Kriz	Consumer	United State	Madison	Wisconsin	53711	Central	OFF-ST-1000	Office Suppli	Storage	Stur-D-Stor S	665.88	6	0	13.3176
18	CA-2014-167164	5/13/14	5/15/14	Second Class	AG-10270	Alejandro Gr	Consumer	United State	West Jordan	Utah	84084	West	OFF-ST-1000	Office Suppli	Storage	Fellowes Sup	55.5	2	0	9.99
19	CA-2014-143336	8/27/14	9/1/14	Second Class	ZD-21925	Zuschuss Dor	Consumer	United State	San Francisc	California	94109	West	OFF-AR-100(	Office Suppli	Art	Newell 341	8.56	2	0	2.4824
20	CA-2014-143336	8/27/14	9/1/14	Second Class	ZD-21925	Zuschuss Dor	Consumer	United State	San Francisc	California	94109	West	TEC-PH-1000	Technology	Phones	Cisco SPA 50	213.48	3	0.2	16.011
21	CA-2014-143336	8/27/14	9/1/14	Second Class	ZD-21925	Zuschuss Dor	Consumer	United State	San Francisc	California	94109	West	OFF-BI-1000	Office Suppli	Binders	Wilson Jones	22.72	4	0.2	7.384
22	CA-2016-137330	12/9/16	12/13/16	Standard Cla	KB-16585	Ken Black	Corporate	United State	Fremont	Nebraska	68025	Central	OFF-AR-100(	Office Suppli	Art	Newell 318	19.46	7	0	5.0596
23	CA-2016-137330	12/9/16	12/13/16	Standard Cla	KB-16585	Ken Black	Corporate	United State	Fremont	Nebraska	68025	Central	OFF-AP-100(	Office Suppli	Appliances	Acco Six-Out	60.34	7	0	15.6884
24	US-2017-156909	7/16/17	7/18/17	Second Class	SF-20065	Sandra Flana	Consumer	United State	Philadelphia	Pennsylvania	19140	East	FUR-CH-100(	Furniture	Chairs	Global Delux	71.372	2	0.3	-1.0196
25	CA-2015-106320	9/25/15	9/30/15	Standard Cla	EB-13870	Emily Burns	Consumer	United State	Orem	Utah	84057	West	FUR-TA-100(	Furniture	Tables	Bretford CR4	1044.63	3	0	240.2649
26	CA-2016-121755	1/16/16	1/20/16	Second Class	EH-13945	Eric Hoffman	Consumer	United State	Los Angeles	California	90049	West	OFF-BI-1000	Office Suppli	Binders	Wilson Jones	11.648	2	0.2	4.2224
27	CA-2016-121755	1/16/16	1/20/16	Second Class	EH-13945	Eric Hoffman	Consumer	United State	Los Angeles	California	90049	West	TEC-AC-1000	Technology	Accessories	Imation 8GB	90.57	3	0	11.7741
28	US-2015-150630	9/17/15	9/21/15	Standard Cla	TB-21520	Tracy Blumst	Consumer	United State	Philadelphia	Pennsylvania	19140	East	FUR-BO-100	Furniture	Bookcases	Riverside Pal	3083.43	7	0.5	-1665.0522
29	US-2015-150630	9/17/15	9/21/15	Standard Cla	TB-21520	Tracy Blumst	Consumer	United State	Philadelphia	Pennsylvania	19140	East	OFF-BI-1000	Office Suppli	Binders	Avery Recycl	9.618	2	0.7	-7.0532
30	US-2015-150630	9/17/15	9/21/15	Standard Cla	TB-21520	Tracy Blumst	Consumer	United State	Philadelphia	Pennsylvania	19140	East	FUR-FU-100(	Furniture	Furnishings	Howard Mill	124.2	3	0.2	15.525
31	US-2015-150630	9/17/15	9/21/15	Standard Cla	TB-21520	Tracy Blumst	Consumer	United State	Philadelphia	Pennsylvania	19140	East	OFF-EN-100(	Office Suppli	Envelopes	Poly String Ti	3.264	2	0.2	1.1016
32	US-2015-150630	9/17/15	9/21/15	Standard Cla	TB-21520	Tracy Blumst	Consumer	United State	Philadelphia	Pennsylvania	19140	East	OFF-AR-100(	Office Suppli	Art	BOSTON Mo	86.304	6	0.2	9.7092
33	US-2015-150630	9/17/15	9/21/15	Standard Cla	TB-21520	Tracy Blumst	Consumer	United State	Philadelphia	Pennsylvania	19140	East	OFF-BI-1000	Office Suppli	Binders	Acco Pressbc	6.858	6	0.7	-5.715

- From the above dataset we can see that, the customer can place multiple orders, the order item will have the product details, so the data can be treated as the following.

Customer Entity: The customer details are stored in it.

Order: Detailed information about an order is provided.

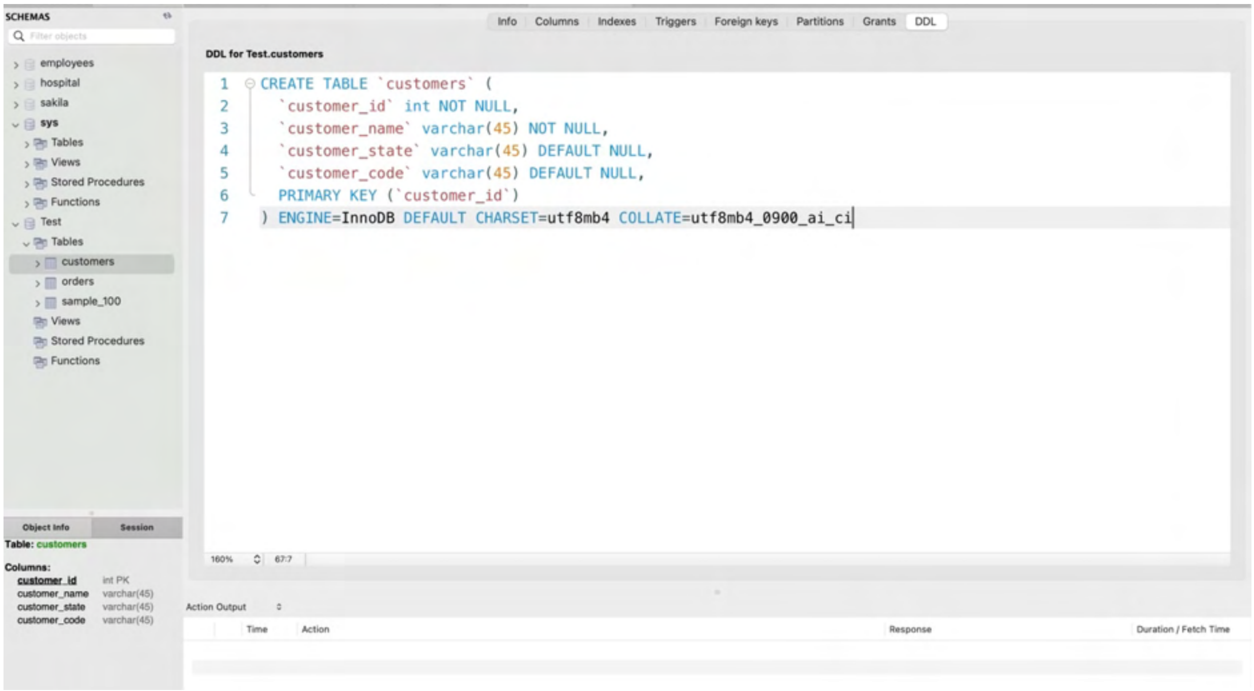
Order item entity: Orders and products will be associated through this table.

Product entity: This field stores information about the product details.

Nikhil Mylarusetty- 016656393

Below are the entities and their records executed.

- Customer id, customer name, customer country, city, state, postal code



DDL for Test.customers

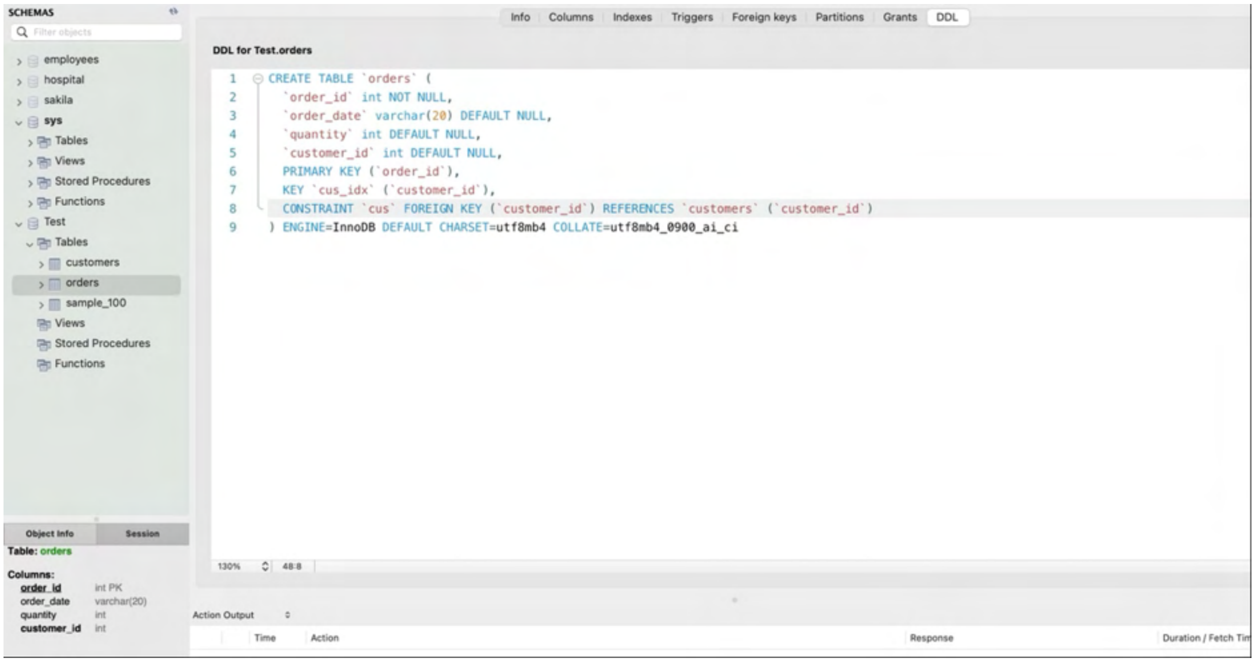
```
CREATE TABLE `customers` (
  `customer_id` int NOT NULL,
  `customer_name` varchar(45) NOT NULL,
  `customer_state` varchar(45) DEFAULT NULL,
  `customer_code` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`customer_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Table: customers

Columns:

customer_id	int PK
customer_name	varchar(45)
customer_state	varchar(45)
customer_code	varchar(45)

- Order id, order date, quantity, and customer ID:



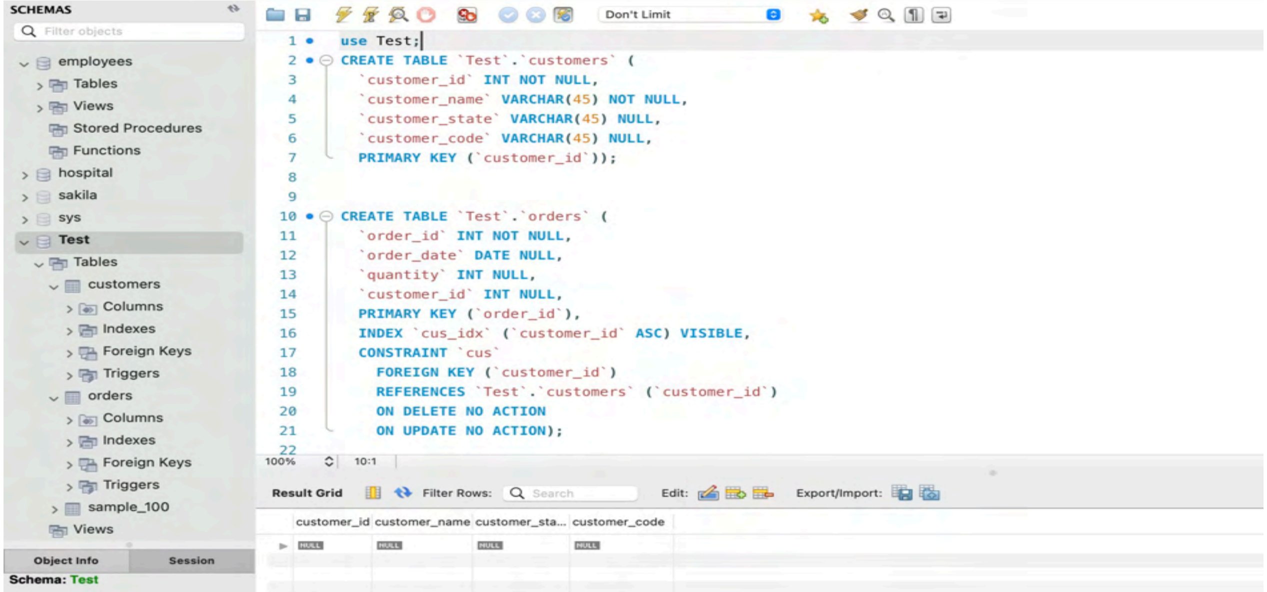
DDL for Test.orders

```
CREATE TABLE `orders` (
  `order_id` int NOT NULL,
  `order_date` varchar(20) DEFAULT NULL,
  `quantity` int DEFAULT NULL,
  `customer_id` int DEFAULT NULL,
  PRIMARY KEY (`order_id`),
  KEY `cus_idx` (`customer_id`),
  CONSTRAINT `cus` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Table: orders

Columns:

order_id	int PK
order_date	varchar(20)
quantity	int
customer_id	int



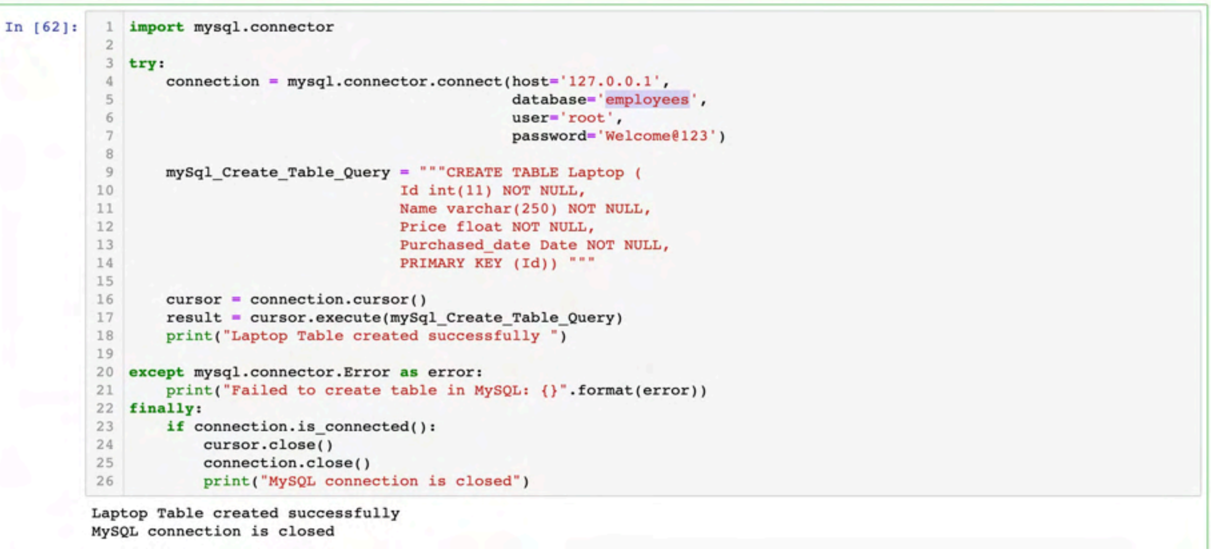
Schema: Test

```sql
use Test;
CREATE TABLE `Test`.`customers` (
  `customer_id` INT NOT NULL,
  `customer_name` VARCHAR(45) NOT NULL,
  `customer_state` VARCHAR(45) NULL,
  `customer_code` VARCHAR(45) NULL,
  PRIMARY KEY (`customer_id`));


CREATE TABLE `Test`.`orders` (
  `order_id` INT NOT NULL,
  `order_date` DATE NULL,
  `quantity` INT NULL,
  `customer_id` INT NULL,
  PRIMARY KEY (`order_id`),
  INDEX `cus_idx` (`customer_id` ASC) VISIBLE,
  CONSTRAINT `cus`
    FOREIGN KEY (`customer_id`)
    REFERENCES `Test`.`customers` (`customer_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

customer_id	customer_name	customer_sta...	customer_code
NULL	NULL	NULL	NULL

c. [10 points]
Write code in Python to perform CRUD operations. The code should include establishing a connection to the MySQL server, executing SQL queries to Create a new database and tables, inserting values from the tables (.csv files) that you came up with in (b) above, Read, Update, and Delete one or more rows.

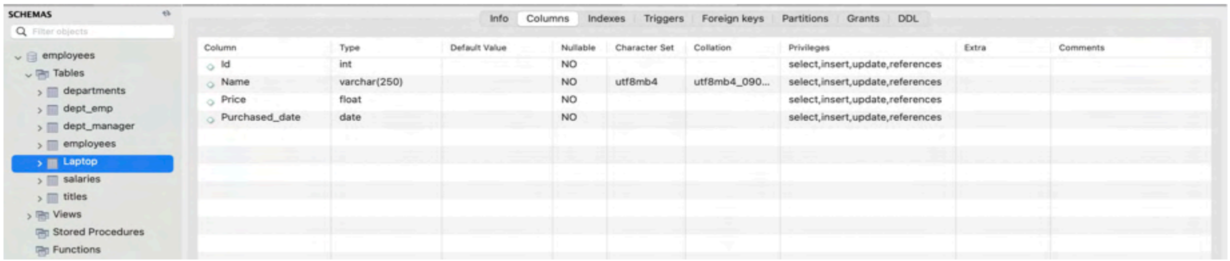
Create: I have successfully created the laptop table in the employee's database. For your reference, here are some images.



In [62]:

```python
import mysql.connector

try:
    connection = mysql.connector.connect(host='127.0.0.1',
                                         database='employees',
                                         user='root',
                                         password='Welcome@123')

    mySql_Create_Table_Query = """CREATE TABLE Laptop (
                             Id int(11) NOT NULL,
                             Name varchar(250) NOT NULL,
                             Price float NOT NULL,
                             Purchased_date Date NOT NULL,
                             PRIMARY KEY (Id)) """

    cursor = connection.cursor()
    result = cursor.execute(mySql_Create_Table_Query)
    print("Laptop Table created successfully ")

except mysql.connector.Error as error:
    print("Failed to create table in MySQL: {}".format(error))
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

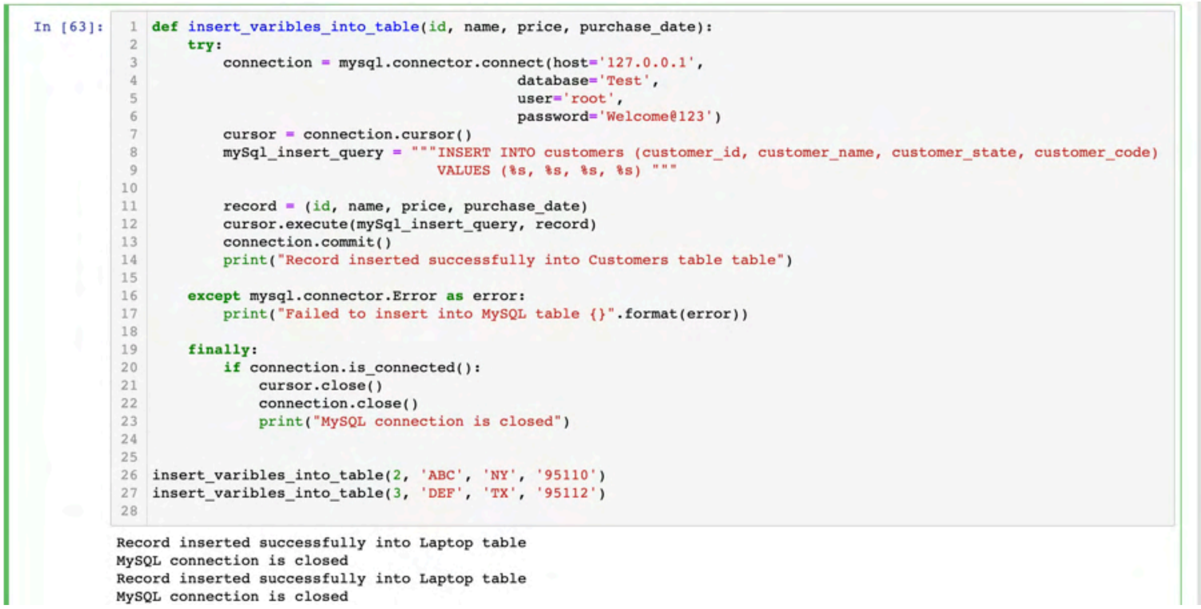
```
Laptop Table created successfully
MySQL connection is closed
```



Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra	Comments
Id	int		NO			select,insert,update,references		
Name	varchar(250)		NO	utf8mb4	utf8mb4_090...	select,insert,update,references		
Price	float		NO			select,insert,update,references		
Purchased_date	date		NO			select,insert,update,references		

Nikhil Mylarusetty- 016656393

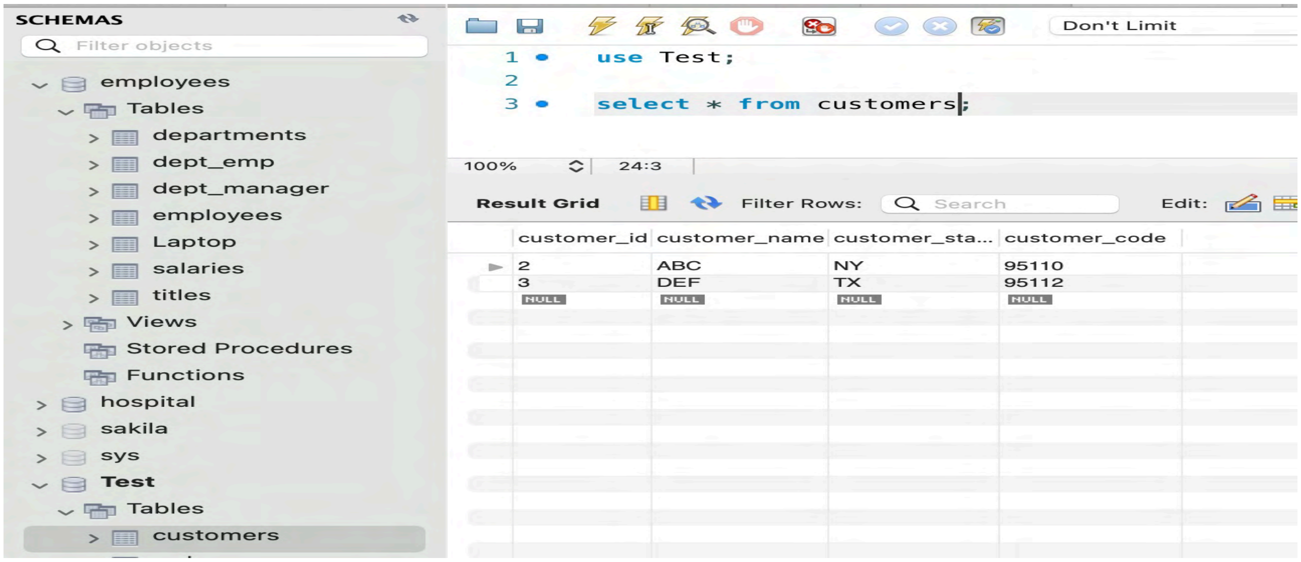
Insert records: Successfully inserted multiple records!



In [63]:

```python
def insert_varibles_into_table(id, name, price, purchase_date):
    try:
        connection = mysql.connector.connect(host='127.0.0.1',
                                             database='Test',
                                             user='root',
                                             password='Welcome@123')
        cursor = connection.cursor()
        mySql_insert_query = """INSERT INTO customers (customer_id, customer_name, customer_state, customer_code)
                                VALUES (%s, %s, %s, %s) """

        record = (id, name, price, purchase_date)
        cursor.execute(mySql_insert_query, record)
        connection.commit()
        print("Record inserted successfully into Customers table table")

    except mysql.connector.Error as error:
        print("Failed to insert into MySQL table {}".format(error))

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("MySQL connection is closed")


insert_varibles_into_table(2, 'ABC', 'NY', '95110')
insert_varibles_into_table(3, 'DEF', 'TX', '95112')

```

```
Record inserted successfully into Laptop table
MySQL connection is closed
Record inserted successfully into Laptop table
MySQL connection is closed
```

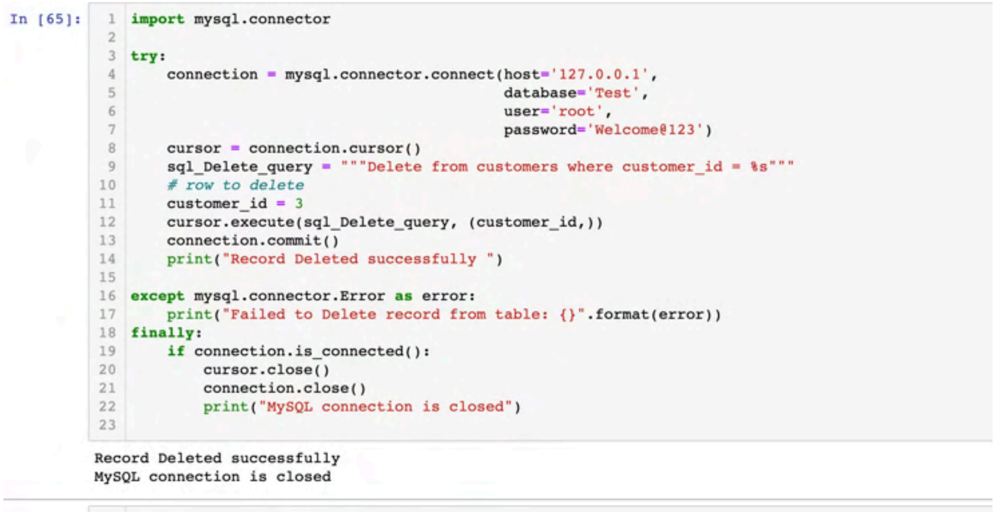


```sql
use Test;

select * from customers;
```

customer_id	customer_name	customer_sta...	customer_code
2	ABC	NY	95110
3	DEF	TX	95112
NULL	NULL	NULL	NULL

Nikhil Mylarusetty- 016656393

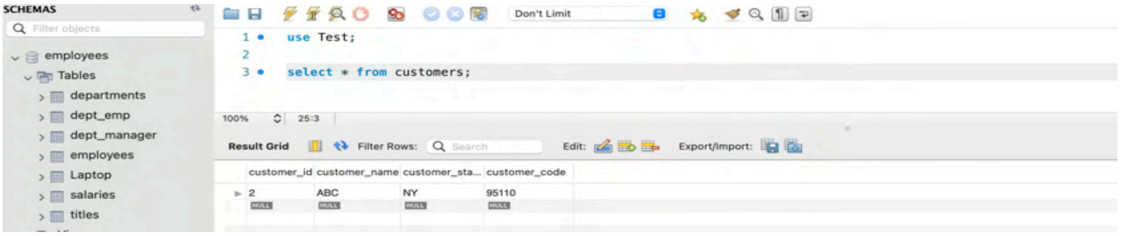
Delete: A successful deletion has been made!



In [65]:

```python
import mysql.connector

try:
    connection = mysql.connector.connect(host='127.0.0.1',
                                         database='Test',
                                         user='root',
                                         password='Welcome@123')

    cursor = connection.cursor()
    sql_Delete_query = """Delete from customers where customer_id = %s"""
    # row to delete
    customer_id = 3
    cursor.execute(sql_Delete_query, (customer_id,))
    connection.commit()
    print("Record Deleted successfully ")

except mysql.connector.Error as error:
    print("Failed to Delete record from table: {}".format(error))
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")

```

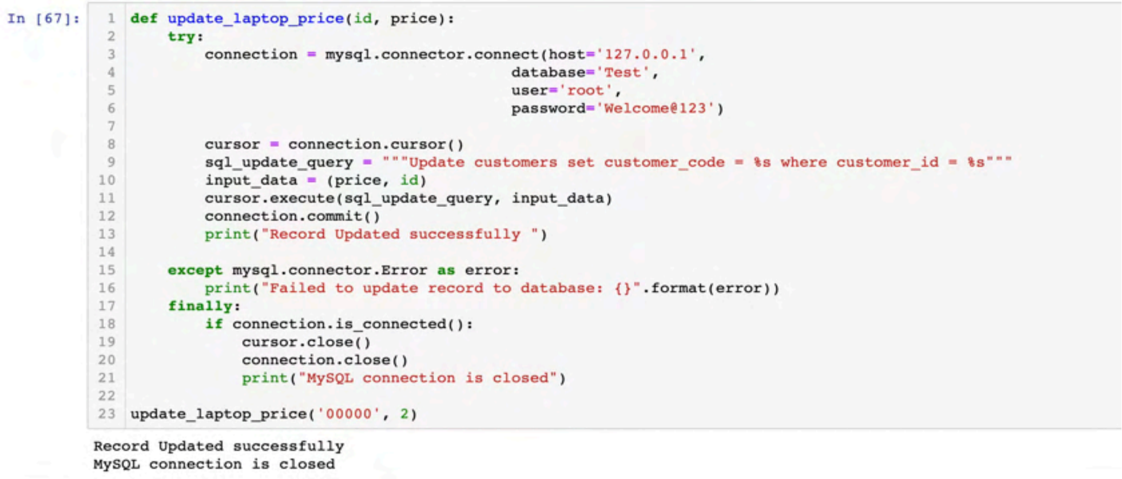
```
Record Deleted successfully
MySQL connection is closed
```



```sql
use Test;

select * from customers;
```

customer_id	customer_name	customer_sta...	customer_code
2	ABC	NY	95110
NULL	NULL	NULL	NULL

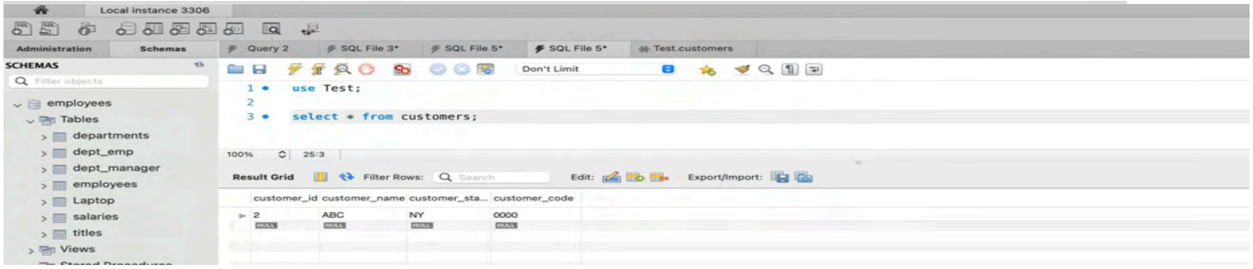
Update: Updated the rows



In [67]:

```python
def update_laptop_price(id, price):
    try:
        connection = mysql.connector.connect(host='127.0.0.1',
                                             database='Test',
                                             user='root',
                                             password='Welcome@123')

        cursor = connection.cursor()
        sql_update_query = """Update customers set customer_code = %s where customer_id = %s"""
        input_data = (price, id)
        cursor.execute(sql_update_query, input_data)
        connection.commit()
        print("Record Updated successfully ")

    except mysql.connector.Error as error:
        print("Failed to update record to database: {}".format(error))
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("MySQL connection is closed")

update_laptop_price('00000', 2)
```

```
Record Updated successfully
MySQL connection is closed
```

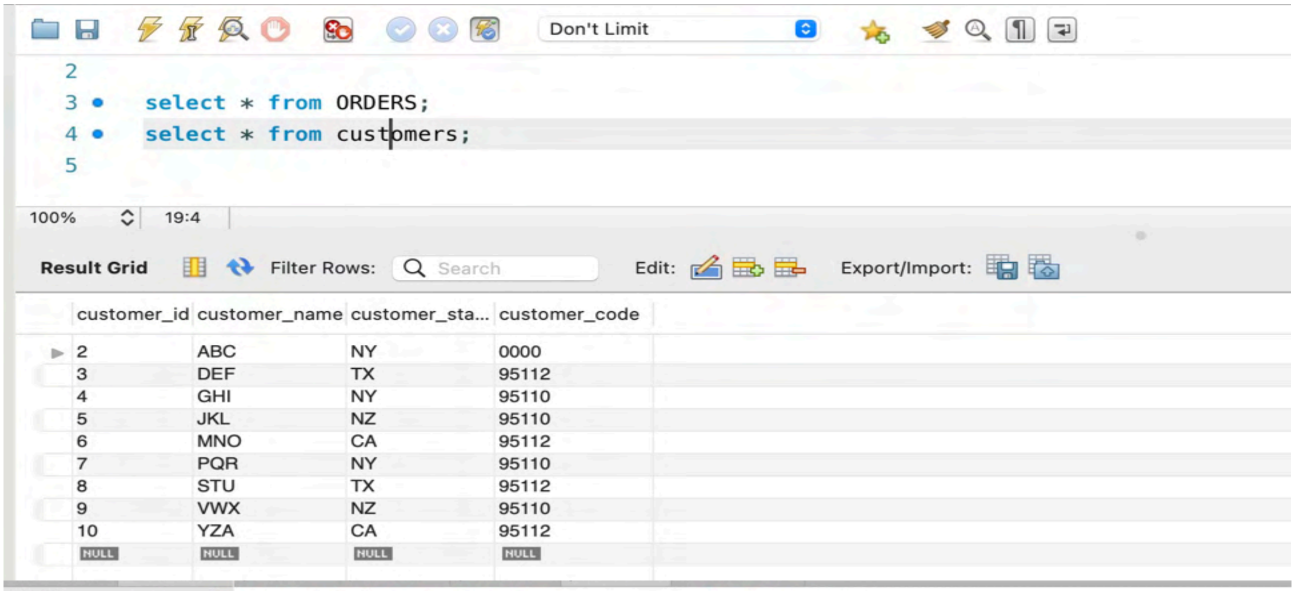


```sql
use Test;

select * from customers;
```

customer_id	customer_name	customer_sta...	customer_code
2	ABC	NY	0000
NULL	NULL	NULL	NULL

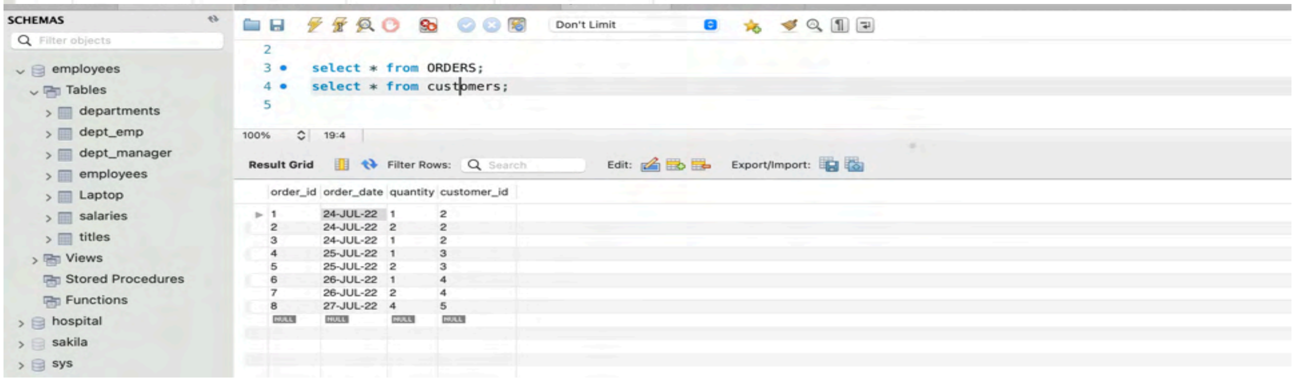
Nikhil Mylarusetty- 016656393

d. [15 points]
In the same python program, exercise some meaningful join statements on the tables created above and using python statements, perform meaningful analysis (one such analysis could be to find the quartiles, standard deviation, outliers, etc. of numerical data) on the records obtained from those SQL statements. From your analysis, draw 5 conclusions that can help the stakeholders (those who will be interested in such conclusions).



```
2
3 • select * from ORDERS;
4 • select * from custpmers;
5
```

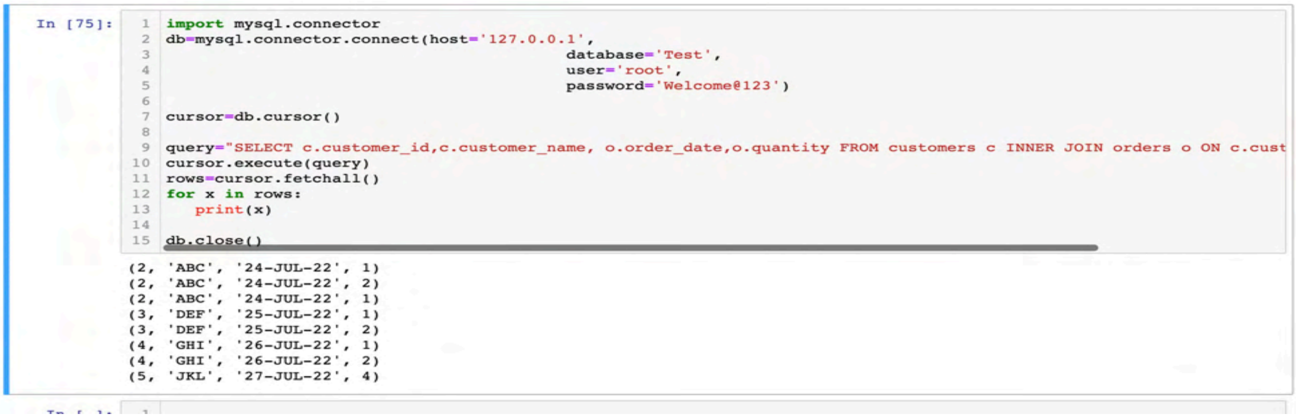
customer_id	customer_name	customer_sta...	customer_code
2	ABC	NY	0000
3	DEF	TX	95112
4	GHI	NY	95110
5	JKL	NZ	95110
6	MNO	CA	95112
7	PQR	NY	95110
8	STU	TX	95112
9	VWX	NZ	95110
10	YZA	CA	95112
NULL	NULL	NULL	NULL



```
2
3 • select * from ORDERS;
4 • select * from custpmers;
5
```

order_id	order_date	quantity	customer_id
1	24-JUL-22	1	2
2	24-JUL-22	2	2
3	24-JUL-22	1	2
4	25-JUL-22	1	3
5	25-JUL-22	2	3
6	26-JUL-22	1	4
7	26-JUL-22	2	4
8	27-JUL-22	4	5
NULL	NULL	NULL	NULL

Inner Join:



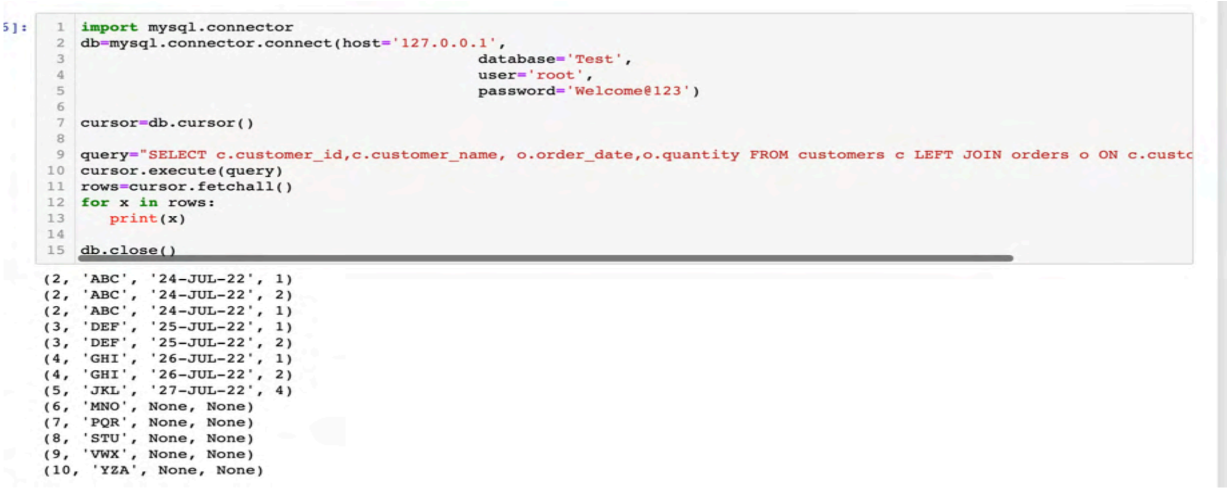
In [75]:

```
import mysql.connector
db=mysql.connector.connect(host='127.0.0.1',
                                                        database='Test',
                                                        user='root',
                                                        password='Welcome@123')

cursor=db.cursor()

query="SELECT c.customer_id,c.customer_name, o.order_date,o.quantity FROM customers c INNER JOIN orders o ON c.cust
cursor.execute(query)
rows=cursor.fetchall()
for x in rows:
    print(x)

db.close()
```

```
(2, 'ABC', '24-JUL-22', 1)
(2, 'ABC', '24-JUL-22', 2)
(2, 'ABC', '24-JUL-22', 1)
(3, 'DEF', '25-JUL-22', 1)
(3, 'DEF', '25-JUL-22', 2)
(4, 'GHI', '26-JUL-22', 1)
(4, 'GHI', '26-JUL-22', 2)
(5, 'JKL', '27-JUL-22', 4)
```

Nikhil Mylarusetty- 016656393

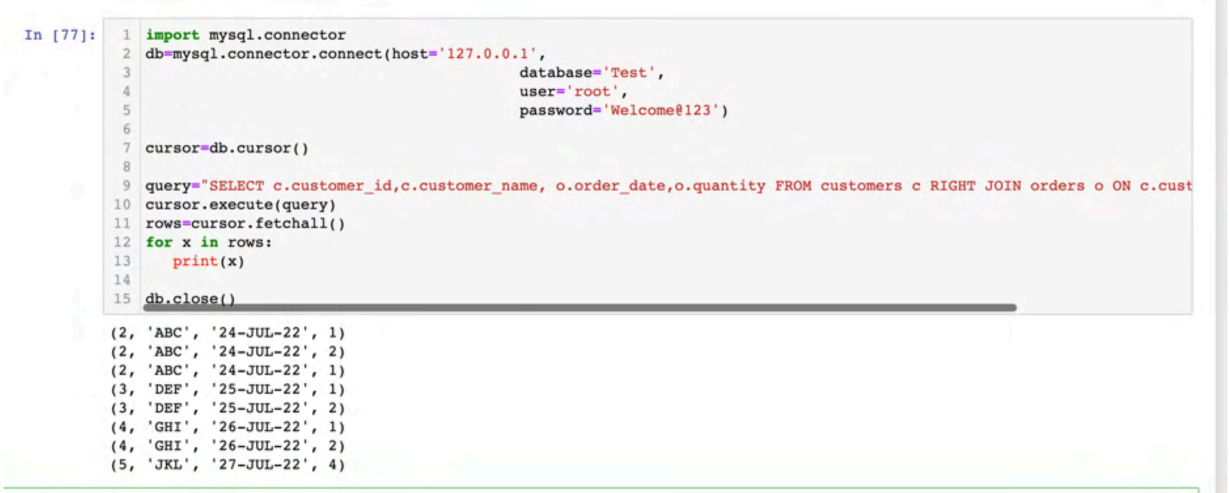
Left Join:



```
import mysql.connector
db=mysql.connector.connect(host='127.0.0.1',
                                              database='Test',
                                              user='root',
                                              password='Welcome@123')

cursor=db.cursor()

query="SELECT c.customer_id,c.customer_name, o.order_date,o.quantity FROM customers c LEFT JOIN orders o ON c.custo
cursor.execute(query)
rows=cursor.fetchall()
for x in rows:
   print(x)

db.close()
```

```
(2, 'ABC', '24-JUL-22', 1)
(2, 'ABC', '24-JUL-22', 2)
(2, 'ABC', '24-JUL-22', 1)
(3, 'DEF', '25-JUL-22', 1)
(3, 'DEF', '25-JUL-22', 2)
(4, 'GHI', '26-JUL-22', 1)
(4, 'GHI', '26-JUL-22', 2)
(5, 'JKL', '27-JUL-22', 4)
(6, 'MNO', None, None)
(7, 'PQR', None, None)
(8, 'STU', None, None)
(9, 'VWX', None, None)
(10, 'YZA', None, None)
```

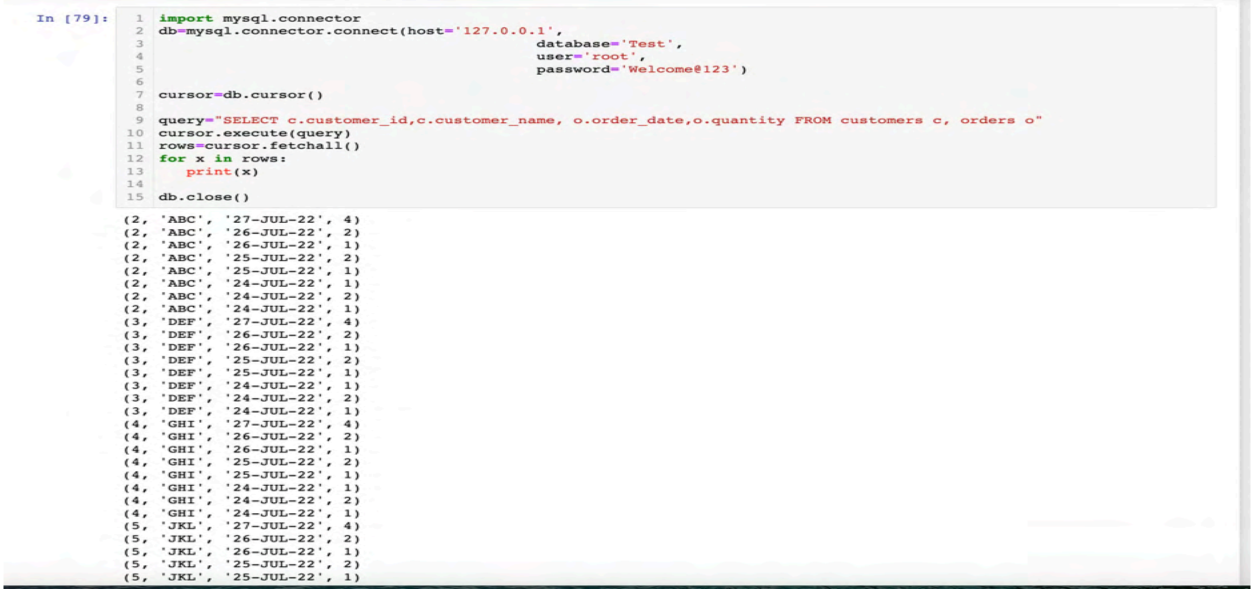
Right Join:



```
import mysql.connector
db=mysql.connector.connect(host='127.0.0.1',
                                              database='Test',
                                              user='root',
                                              password='Welcome@123')

cursor=db.cursor()

query="SELECT c.customer_id,c.customer_name, o.order_date,o.quantity FROM customers c RIGHT JOIN orders o ON c.cust
cursor.execute(query)
rows=cursor.fetchall()
for x in rows:
   print(x)

db.close()
```

```
(2, 'ABC', '24-JUL-22', 1)
(2, 'ABC', '24-JUL-22', 2)
(2, 'ABC', '24-JUL-22', 1)
(3, 'DEF', '25-JUL-22', 1)
(3, 'DEF', '25-JUL-22', 2)
(4, 'GHI', '26-JUL-22', 1)
(4, 'GHI', '26-JUL-22', 2)
(5, 'JKL', '27-JUL-22', 4)
```

Cross Join:

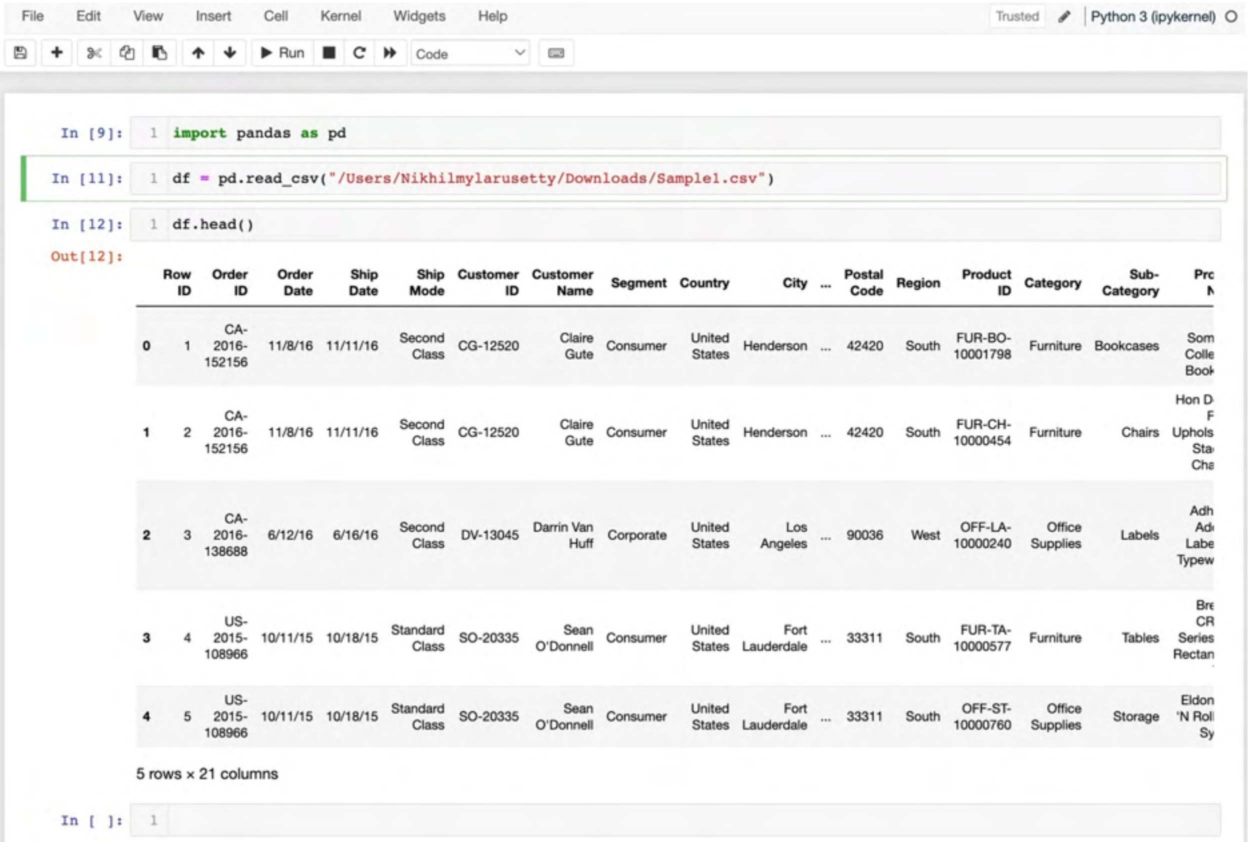


```
import mysql.connector
db=mysql.connector.connect(host='127.0.0.1',
                                              database='Test',
                                              user='root',
                                              password='Welcome@123')

cursor=db.cursor()

query="SELECT c.customer_id,c.customer_name, o.order_date,o.quantity FROM customers c, orders o"
cursor.execute(query)
rows=cursor.fetchall()
for x in rows:
   print(x)

db.close()
```

```
(2, 'ABC', '27-JUL-22', 4)
(2, 'ABC', '26-JUL-22', 2)
(2, 'ABC', '26-JUL-22', 1)
(2, 'ABC', '25-JUL-22', 2)
(2, 'ABC', '25-JUL-22', 1)
(2, 'ABC', '24-JUL-22', 1)
(2, 'ABC', '24-JUL-22', 2)
(2, 'ABC', '24-JUL-22', 1)
(3, 'DEF', '27-JUL-22', 4)
(3, 'DEF', '26-JUL-22', 2)
(3, 'DEF', '26-JUL-22', 1)
(3, 'DEF', '25-JUL-22', 2)
(3, 'DEF', '25-JUL-22', 1)
(3, 'DEF', '24-JUL-22', 1)
(3, 'DEF', '24-JUL-22', 2)
(3, 'DEF', '24-JUL-22', 1)
(4, 'GHI', '27-JUL-22', 4)
(4, 'GHI', '26-JUL-22', 2)
(4, 'GHI', '26-JUL-22', 1)
(4, 'GHI', '25-JUL-22', 2)
(4, 'GHI', '25-JUL-22', 1)
(4, 'GHI', '24-JUL-22', 1)
(4, 'GHI', '24-JUL-22', 2)
(4, 'GHI', '24-JUL-22', 1)
(5, 'JKL', '27-JUL-22', 4)
(5, 'JKL', '26-JUL-22', 2)
(5, 'JKL', '26-JUL-22', 1)
(5, 'JKL', '25-JUL-22', 2)
(5, 'JKL', '25-JUL-22', 1)
```

Nikhil Mylarusetty- 016656393

Key Insights and some analysis on the data: I Choose profit column as the numerical column because it is the most insightful one from the entire dataset.



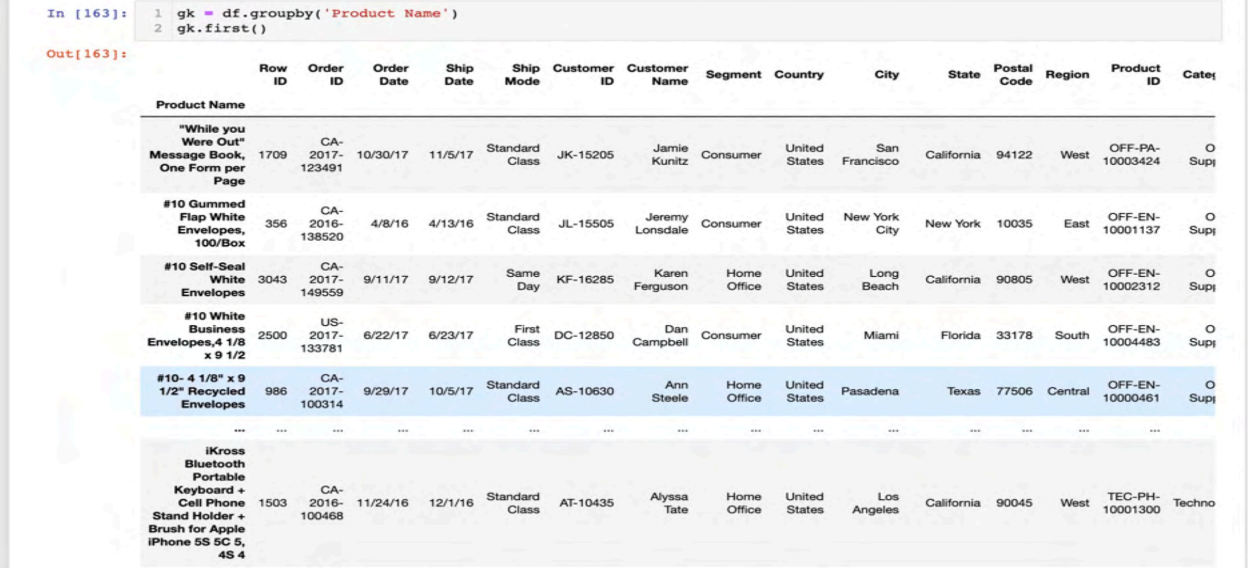
```
In [9]:  import pandas as pd

In [11]: df = pd.read_csv("/Users/Nikhilmylarusetty/Downloads/Sample1.csv")

In [12]: df.head()

Out[12]:
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Pro N
0	1	CA-2016-152156	11/8/16	11/11/16	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bookcases	Som Colle Book
1	2	CA-2016-152156	11/8/16	11/11/16	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon D F Uphols Sta Cha
2	3	CA-2016-138688	6/12/16	6/16/16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	Labels	Adh Ad Labe Typew
3	4	US-2015-108966	10/11/15	10/18/15	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	Tables	Bre CR Series Rectan
4	5	US-2015-108966	10/11/15	10/18/15	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon 'N Roll Sy

5 rows × 21 columns

```
In [ ]:
```

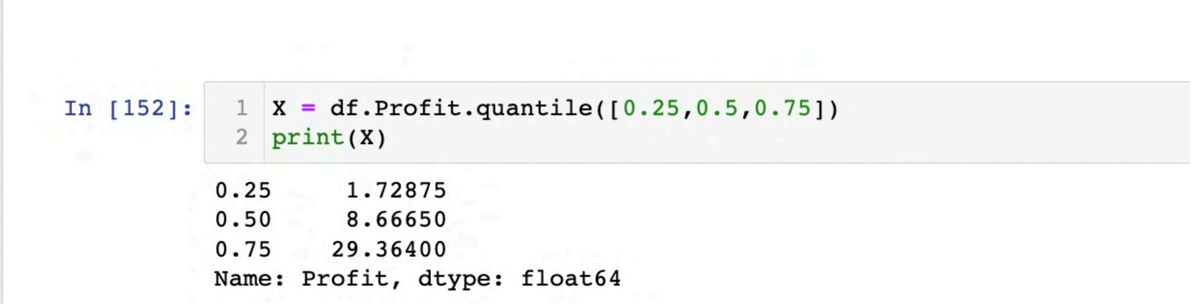
- To determine the profit and loss information for each group, I categorized the data using the group by function.



```
In [163]: gk = df.groupby('Product Name')
          gk.first()

Out[163]:
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Categ
Product Name															
"While you Were Out" Message Book, One Form per Page	1709	CA-2017-123491	10/30/17	11/5/17	Standard Class	JK-15205	Jamie Kunitz	Consumer	United States	San Francisco	California	94122	West	OFF-PA-10003424	O Sup
#10 Gummed Flap White Envelopes, 100/Box	356	CA-2016-138520	4/8/16	4/13/16	Standard Class	JL-15505	Jeremy Lonsdale	Consumer	United States	New York City	New York	10035	East	OFF-EN-10001137	O Sup
#10 Self-Seal White Envelopes	3043	CA-2017-149559	9/11/17	9/12/17	Same Day	KF-16285	Karen Ferguson	Home Office	United States	Long Beach	California	90805	West	OFF-EN-10002312	O Sup
#10 White Business Envelopes,4 1/8 x 9 1/2	2500	US-2017-133781	6/22/17	6/23/17	First Class	DC-12850	Dan Campbell	Consumer	United States	Miami	Florida	33178	South	OFF-EN-10004483	O Sup
#10- 4 1/8" x 9 1/2" Recycled Envelopes	986	CA-2017-100314	9/29/17	10/5/17	Standard Class	AS-10630	Ann Steele	Home Office	United States	Pasadena	Texas	77506	Central	OFF-EN-10000461	O Sup
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
iKross Bluetooth Portable Keyboard + Cell Phone Stand Holder + Brush for Apple iPhone 5S 5C 5, 4S 4	1503	CA-2016-100468	11/24/16	12/1/16	Standard Class	AT-10435	Alyssa Tate	Home Office	United States	Los Angeles	California	90045	West	TEC-PH-10001300	Techno

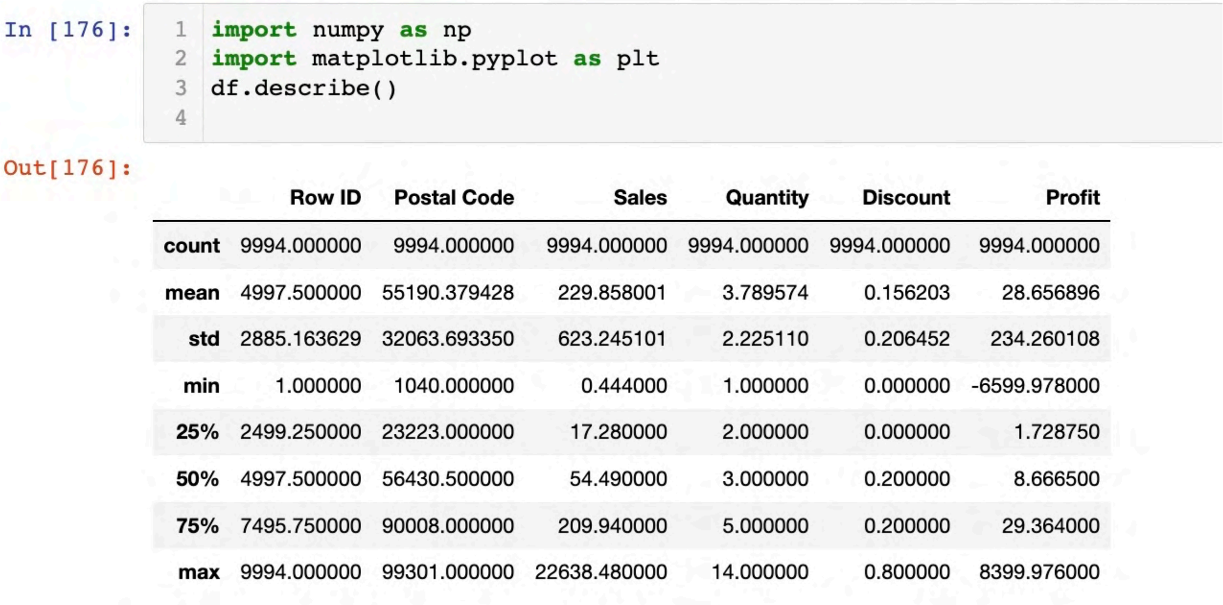
- As part of the quantitative analysis, we also obtained all percentile values for the profit column. This gives us a deeper understanding of the data.



In [152]:
```
X = df.Profit.quantile([0.25,0.5,0.75])
print(X)
```

```
0.25     1.72875
0.50     8.66650
0.75    29.36400
Name: Profit, dtype: float64
```

- The following figure displays the overall profit generated from the entire dataset along with the minimum and maximum values.
- Standard deviation is used to measure the amount of data that is centered around a mean value.
- Generally, box plots are used to eliminate the anomalies/outliers, but in this case as temperature is the key factor, we cannot eliminate the anomalies/ outliers.



In [176]:
```
import numpy as np
import matplotlib.pyplot as plt
df.describe()

```

Out[176]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28.656896
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29.364000
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399.976000

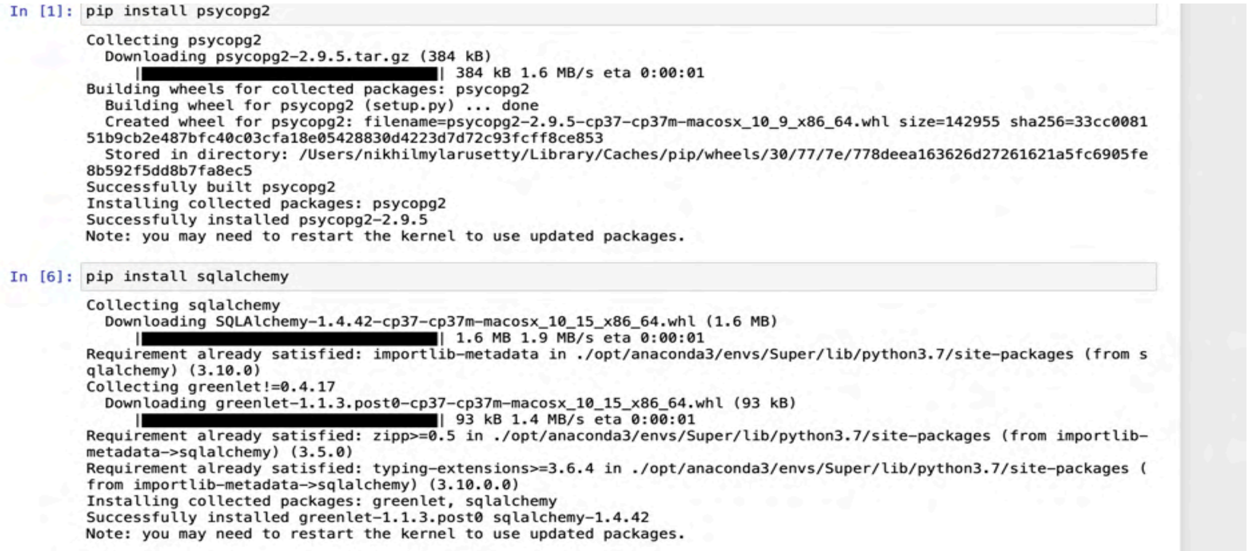
- Analyzing the dataset carefully, we can conclude that Canon Image Advanced Copier generates the highest profit of all the categories
- Furthermore, profit is largely dependent on state and city characteristics. As we can see that state and city profits are primarily coming from California, Texas, and New York.

Nikhil Mylarusetty- 016656393

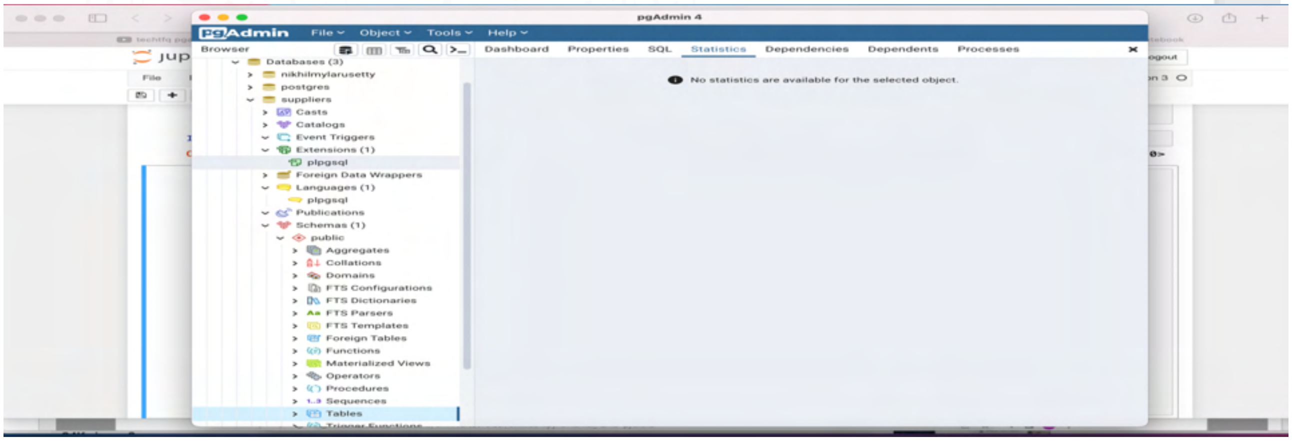
2. [15 points] Python with Postgres
   Repeat the above steps 1a and 1c for the same dataset and same CRUD operations with Postgres, Python, and psycopg2. Please feel free to refer

   to https://www.postgresqltutorial.com/postgresql-python/ Links to an external site. for guidance.
   Alternatively, you can also use SQLAlchemy. Please feel free to refer to the following tutorials for guidance in that case: https://realpython.com/flask-by-example-part-2-postgres-sqlalchemy-and-alembic/ Links to an external site. and https://www.learndatasci.com/tutorials/using-databases-python-postgres-sqlalchemy-and-alembic/ Links to an external site. (No need to do any migrations or alembic; just CRUD operations).

Installed PostgreSQL Successfully:

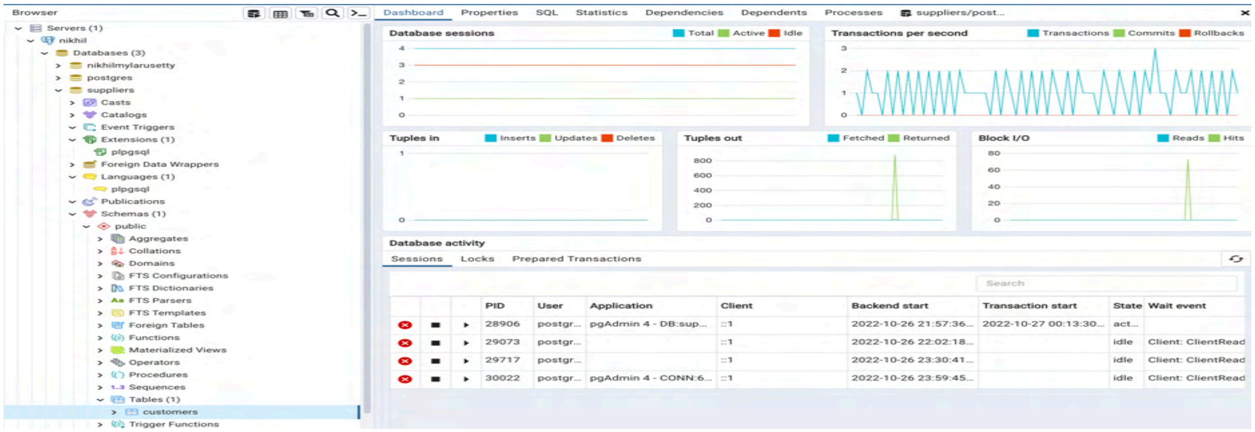


```
In [1]: pip install psycopg2

        Collecting psycopg2
          Downloading psycopg2-2.9.5.tar.gz (384 kB)
             |████████████████████████████████| 384 kB 1.6 MB/s eta 0:00:01
        Building wheels for collected packages: psycopg2
          Building wheel for psycopg2 (setup.py) ... done
          Created wheel for psycopg2: filename=psycopg2-2.9.5-cp37-cp37m-macosx_10_9_x86_64.whl size=142955 sha256=33cc0081
        51b9cb2e487bfc40c03cfa18e05428830d4223d7d72c93fcff8ce853
          Stored in directory: /Users/nikhilmylarusetty/Library/Caches/pip/wheels/30/77/7e/778deea163626d27261621a5fc6905fe
        8b592f5dd8b7fa8ec5
        Successfully built psycopg2
        Installing collected packages: psycopg2
        Successfully installed psycopg2-2.9.5
        Note: you may need to restart the kernel to use updated packages.

In [6]: pip install sqlalchemy

        Collecting sqlalchemy
          Downloading SQLAlchemy-1.4.42-cp37-cp37m-macosx_10_15_x86_64.whl (1.6 MB)
             |████████████████████████████████| 1.6 MB 1.9 MB/s eta 0:00:01
        Requirement already satisfied: importlib-metadata in ./opt/anaconda3/envs/Super/lib/python3.7/site-packages (from s
        qlalchemy) (3.10.0)
        Collecting greenlet!=0.4.17
          Downloading greenlet-1.1.3.post0-cp37-cp37m-macosx_10_15_x86_64.whl (93 kB)
             |████████████████████████████████| 93 kB 1.4 MB/s eta 0:00:01
        Requirement already satisfied: zipp>=0.5 in ./opt/anaconda3/envs/Super/lib/python3.7/site-packages (from importlib-
        metadata->sqlalchemy) (3.5.0)
        Requirement already satisfied: typing-extensions>=3.6.4 in ./opt/anaconda3/envs/Super/lib/python3.7/site-packages (
        from importlib-metadata->sqlalchemy) (3.10.0.0)
        Installing collected packages: greenlet, sqlalchemy
        Successfully installed greenlet-1.1.3.post0 sqlalchemy-1.4.42
        Note: you may need to restart the kernel to use updated packages.
```



No statistics are available for the selected object.

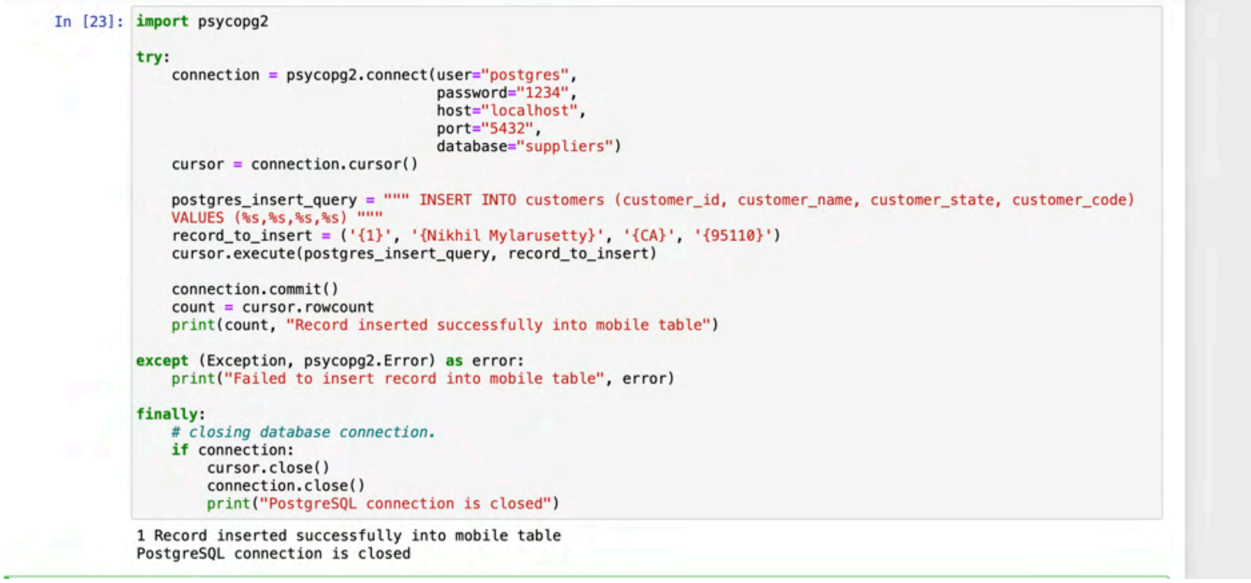
Nikhil Mylarusetty- 016656393



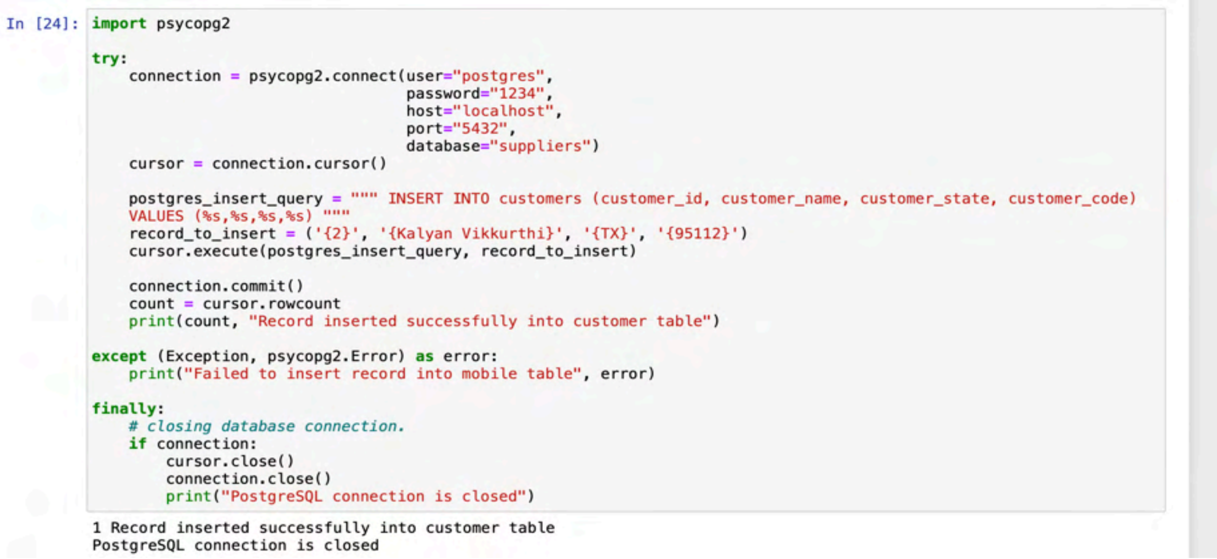
			PID	User	Application	Client	Backend start	Transaction start	State	Wait event
⊗	■	▸	28906	postgr...	pgAdmin 4 - DB:sup...	::1	2022-10-26 21:57:36...	2022-10-27 00:13:30...	act...	
⊗	■	▸	29073	postgr...		::1	2022-10-26 22:02:18...		idle	Client: ClientRead
⊗	■	▸	29717	postgr...		::1	2022-10-26 23:30:41...		idle	Client: ClientRead
⊗	■	▸	30022	postgr...	pgAdmin 4 - CONN:6...	::1	2022-10-26 23:59:45...		idle	Client: ClientRead

Performed CRUD operations on the database employees

Insertion:



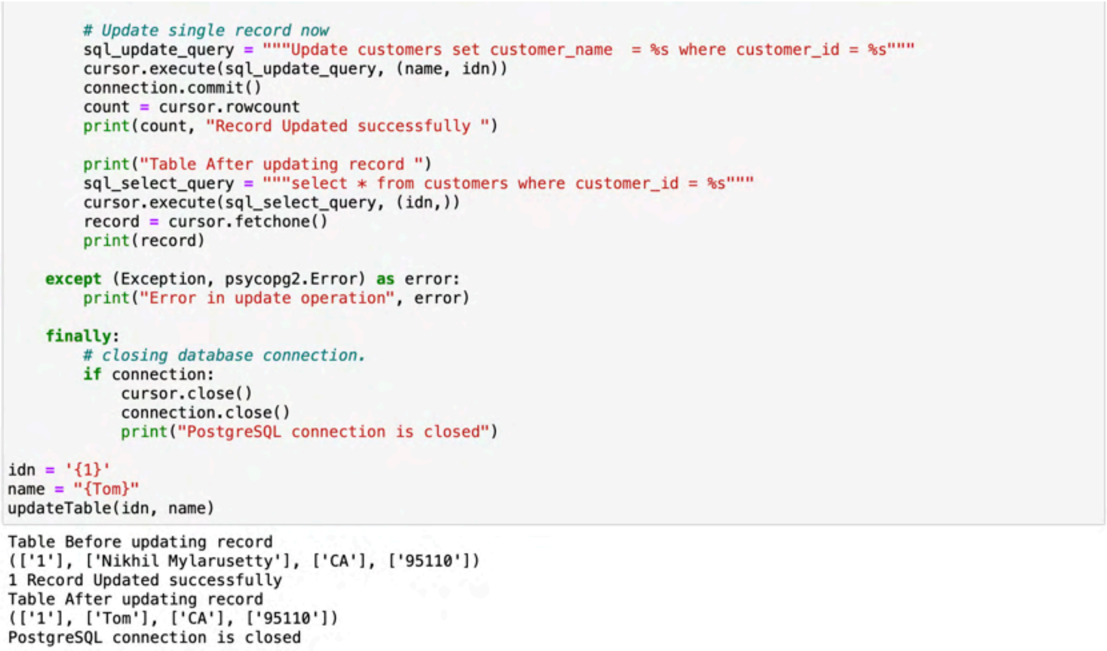
```
In [23]: import psycopg2

         try:
             connection = psycopg2.connect(user="postgres",
                                           password="1234",
                                           host="localhost",
                                           port="5432",
                                           database="suppliers")
             cursor = connection.cursor()

             postgres_insert_query = """ INSERT INTO customers (customer_id, customer_name, customer_state, customer_code)
             VALUES (%s,%s,%s,%s) """
             record_to_insert = ('{1}', '{Nikhil Mylarusetty}', '{CA}', '{95110}')
             cursor.execute(postgres_insert_query, record_to_insert)

             connection.commit()
             count = cursor.rowcount
             print(count, "Record inserted successfully into mobile table")

         except (Exception, psycopg2.Error) as error:
             print("Failed to insert record into mobile table", error)

         finally:
             # closing database connection.
             if connection:
                 cursor.close()
                 connection.close()
                 print("PostgreSQL connection is closed")

         1 Record inserted successfully into mobile table
         PostgreSQL connection is closed
```



```
In [24]: import psycopg2

         try:
             connection = psycopg2.connect(user="postgres",
                                           password="1234",
                                           host="localhost",
                                           port="5432",
                                           database="suppliers")
             cursor = connection.cursor()

             postgres_insert_query = """ INSERT INTO customers (customer_id, customer_name, customer_state, customer_code)
             VALUES (%s,%s,%s,%s) """
             record_to_insert = ('{2}', '{Kalyan Vikkurthi}', '{TX}', '{95112}')
             cursor.execute(postgres_insert_query, record_to_insert)

             connection.commit()
             count = cursor.rowcount
             print(count, "Record inserted successfully into customer table")

         except (Exception, psycopg2.Error) as error:
             print("Failed to insert record into mobile table", error)

         finally:
             # closing database connection.
             if connection:
                 cursor.close()
                 connection.close()
                 print("PostgreSQL connection is closed")

         1 Record inserted successfully into customer table
         PostgreSQL connection is closed
```

Nikhil Mylarusetty- 016656393

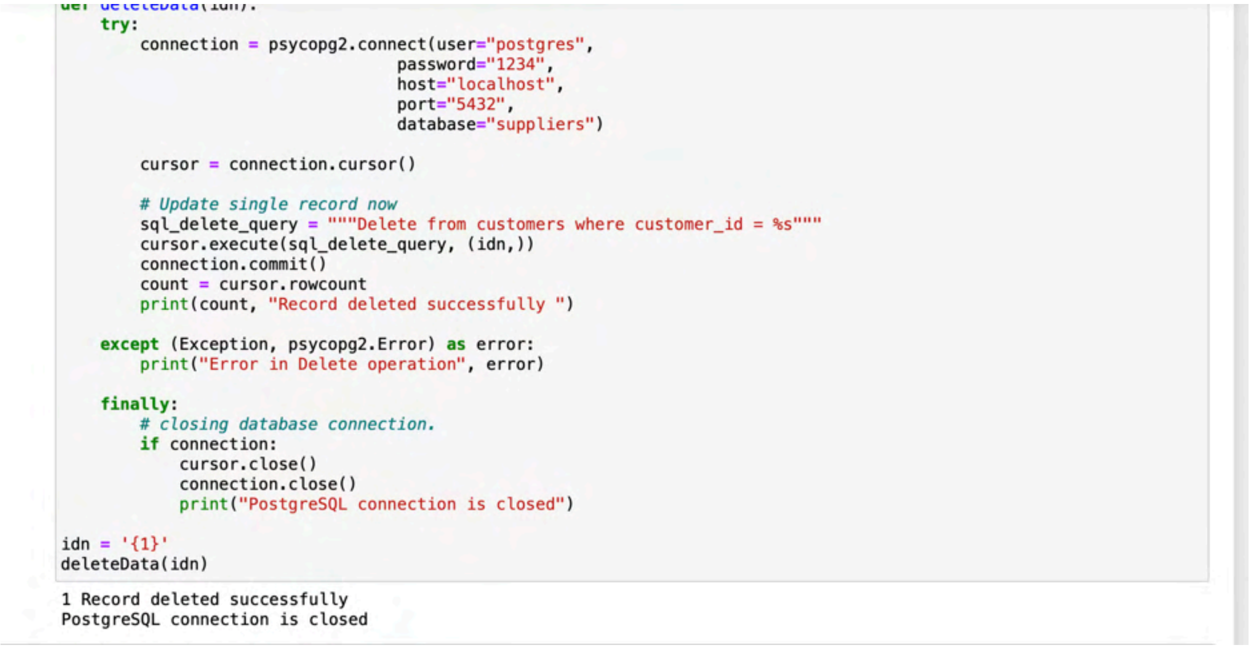
Update and Read:



```
        # Update single record now
        sql_update_query = """Update customers set customer_name  = %s where customer_id = %s"""
        cursor.execute(sql_update_query, (name, idn))
        connection.commit()
        count = cursor.rowcount
        print(count, "Record Updated successfully ")

        print("Table After updating record ")
        sql_select_query = """select * from customers where customer_id = %s"""
        cursor.execute(sql_select_query, (idn,))
        record = cursor.fetchone()
        print(record)

    except (Exception, psycopg2.Error) as error:
        print("Error in update operation", error)

    finally:
        # closing database connection.
        if connection:
            cursor.close()
            connection.close()
            print("PostgreSQL connection is closed")

idn = '{1}'
name = "{Tom}"
updateTable(idn, name)
```

```
Table Before updating record
(['1'], ['Nikhil Mylarusetty'], ['CA'], ['95110'])
1 Record Updated successfully
Table After updating record
(['1'], ['Tom'], ['CA'], ['95110'])
PostgreSQL connection is closed
```

Delete:



```
def deleteData(idn):
    try:
        connection = psycopg2.connect(user="postgres",
                                  password="1234",
                                  host="localhost",
                                  port="5432",
                                  database="suppliers")

        cursor = connection.cursor()

        # Update single record now
        sql_delete_query = """Delete from customers where customer_id = %s"""
        cursor.execute(sql_delete_query, (idn,))
        connection.commit()
        count = cursor.rowcount
        print(count, "Record deleted successfully ")

    except (Exception, psycopg2.Error) as error:
        print("Error in Delete operation", error)

    finally:
        # closing database connection.
        if connection:
            cursor.close()
            connection.close()
            print("PostgreSQL connection is closed")

idn = '{1}'
deleteData(idn)
```

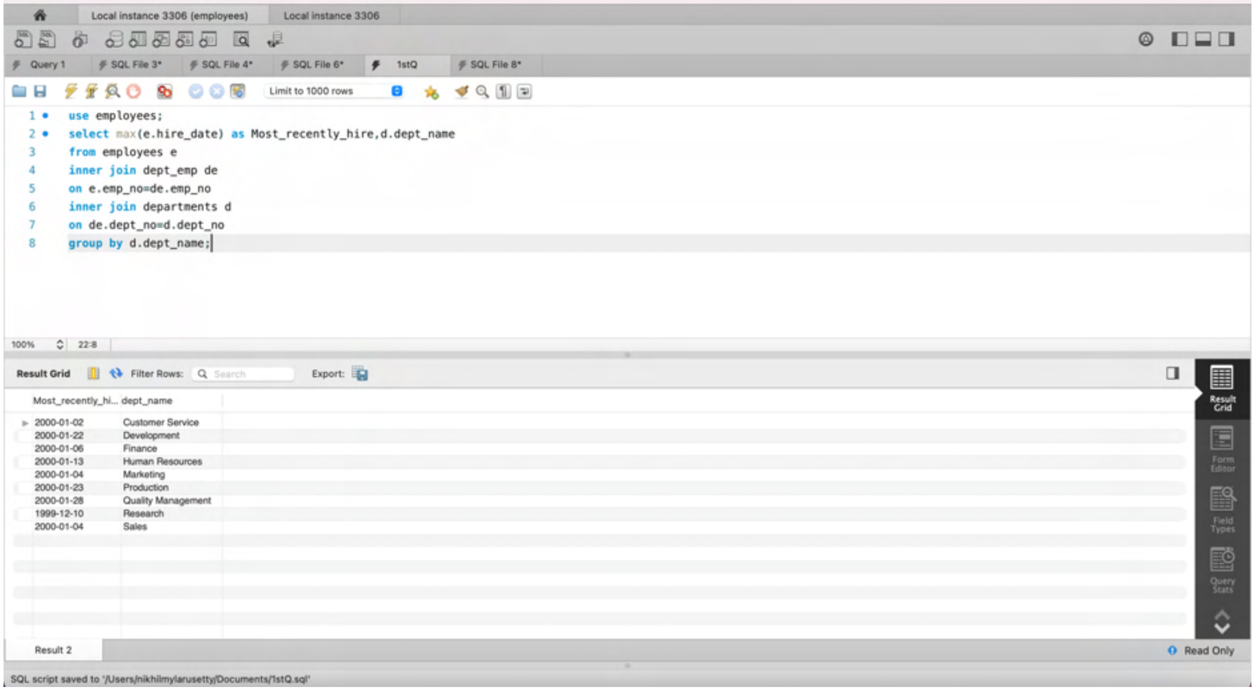
```
1 Record deleted successfully
PostgreSQL connection is closed
```

Nikhil Mylarusetty- 016656393

3. [30 points]

From the employees database that you imported into MySQL in your earlier HW, write SQL queries to get the following information and show sample (or full, if it is small) output for each. If it is not possible to implement them in SQL, state the same.

I. The most recently hired employee in each of the departments of the organization

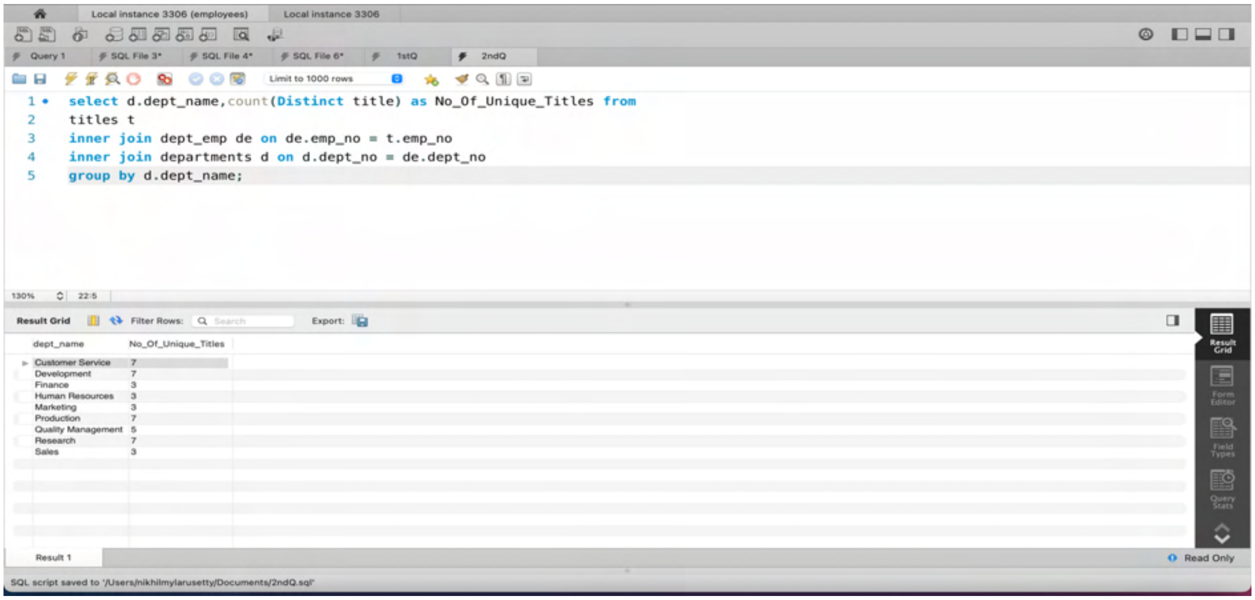


```sql
use employees;
select max(e.hire_date) as Most_recently_hire,d.dept_name
from employees e
inner join dept_emp de
on e.emp_no=de.emp_no
inner join departments d
on de.dept_no=d.dept_no
group by d.dept_name;
```

Most_recently_hi...	dept_name
2000-01-02	Customer Service
2000-01-22	Development
2000-01-06	Finance
2000-01-13	Human Resources
2000-01-04	Marketing
2000-01-23	Production
2000-01-28	Quality Management
1999-12-10	Research
2000-01-04	Sales

SQL script saved to '/Users/nikhilmylarusetty/Documents/1stQ.sql'

II. The number of unique titles in each department



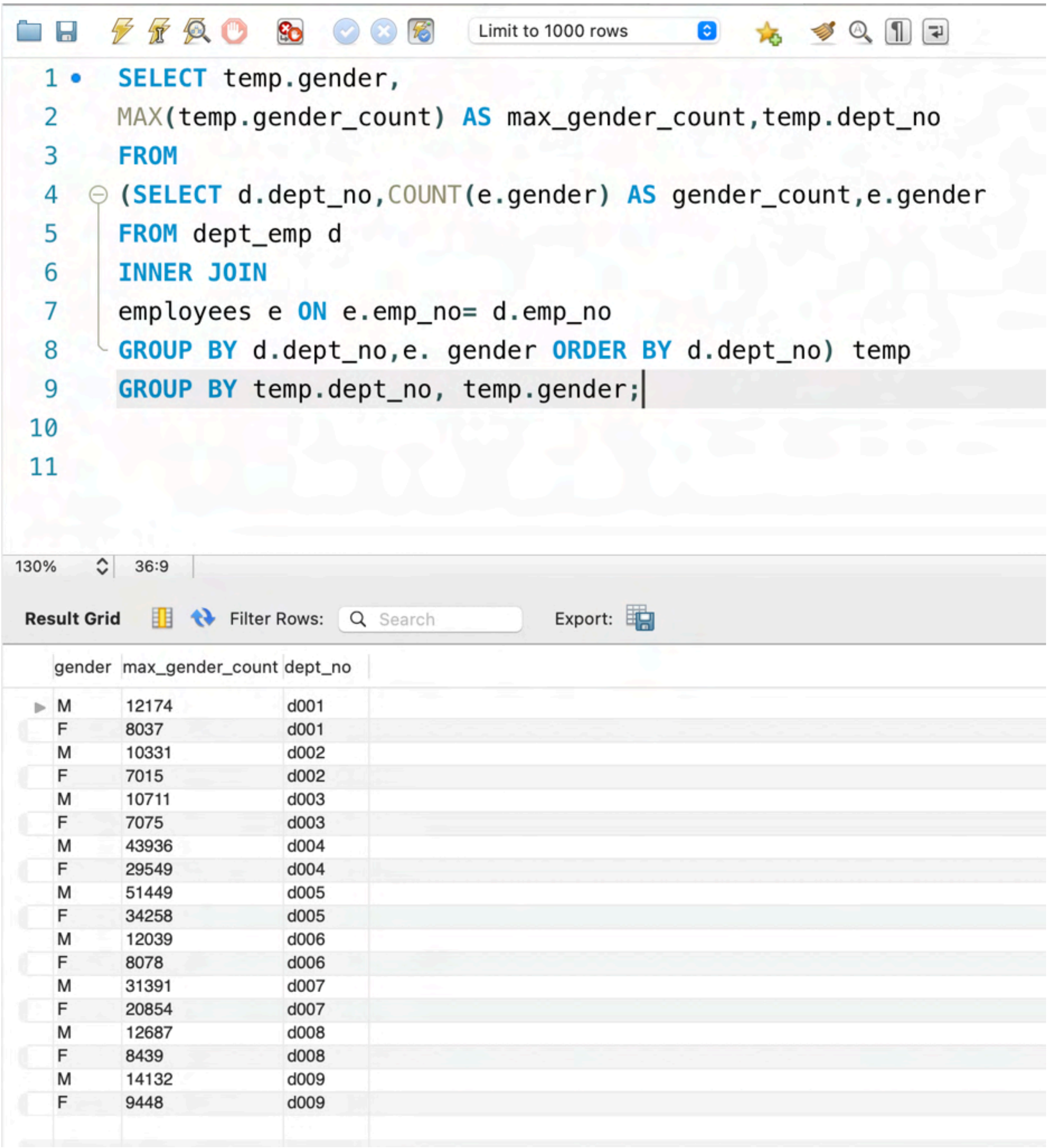
```sql
select d.dept_name,count(Distinct title) as No_Of_Unique_Titles from
titles t
inner join dept_emp de on de.emp_no = t.emp_no
inner join departments d on d.dept_no = de.dept_no
group by d.dept_name;
```

dept_name	No_Of_Unique_Titles
Customer Service	7
Development	7
Finance	3
Human Resources	3
Marketing	3
Production	7
Quality Management	5
Research	7
Sales	3

SQL script saved to '/Users/nikhilmylarusetty/Documents/2ndQ.sql'

Nikhil Mylarusetty- 016656393

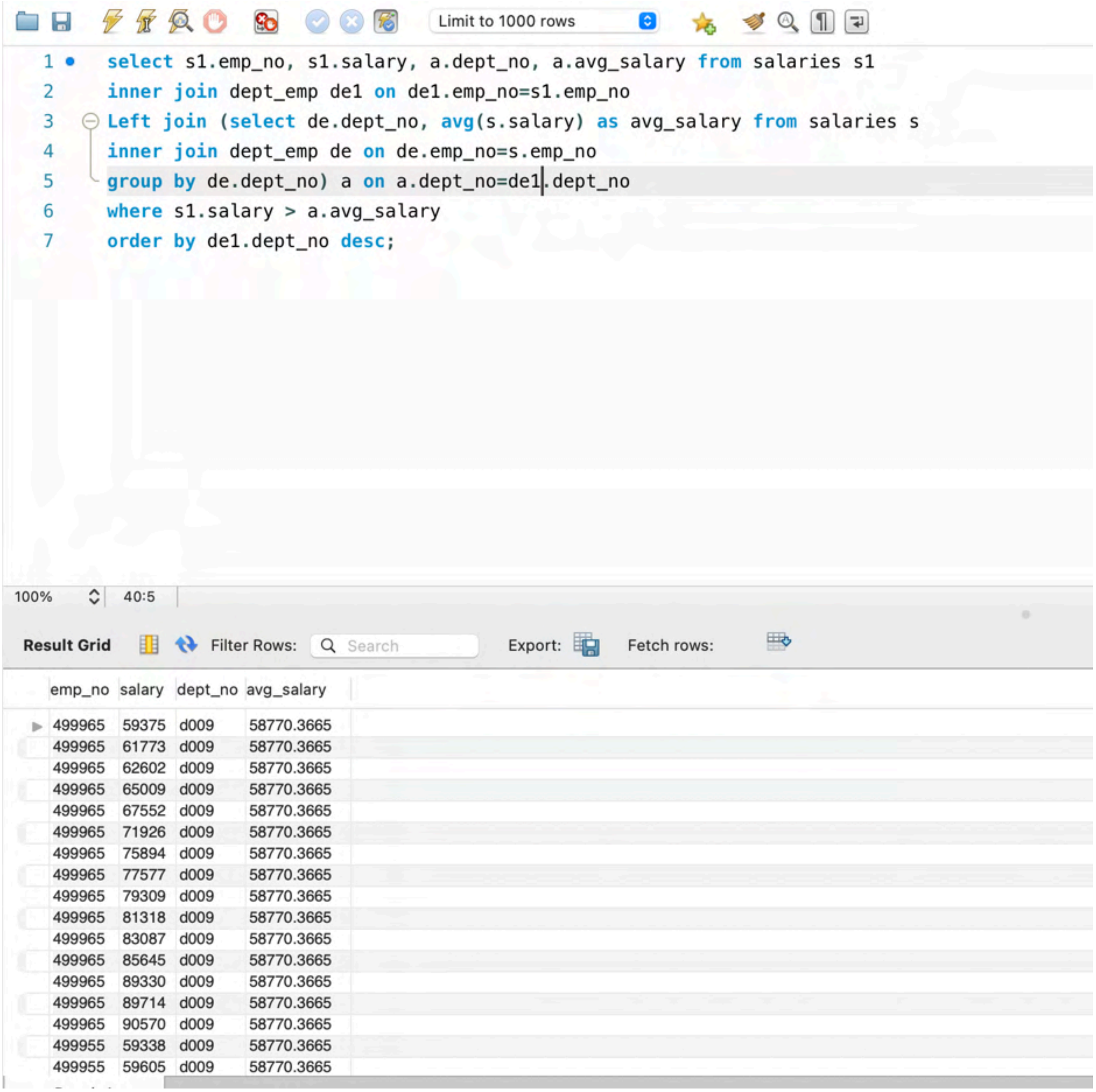
III. The majority gender in each department



```sql
SELECT temp.gender,
MAX(temp.gender_count) AS max_gender_count,temp.dept_no
FROM
(SELECT d.dept_no,COUNT(e.gender) AS gender_count,e.gender
FROM dept_emp d
INNER JOIN
employees e ON e.emp_no= d.emp_no
GROUP BY d.dept_no,e. gender ORDER BY d.dept_no) temp
GROUP BY temp.dept_no, temp.gender;
```

gender	max_gender_count	dept_no
M	12174	d001
F	8037	d001
M	10331	d002
F	7015	d002
M	10711	d003
F	7075	d003
M	43936	d004
F	29549	d004
M	51449	d005
F	34258	d005
M	12039	d006
F	8078	d006
M	31391	d007
F	20854	d007
M	12687	d008
F	8439	d008
M	14132	d009
F	9448	d009

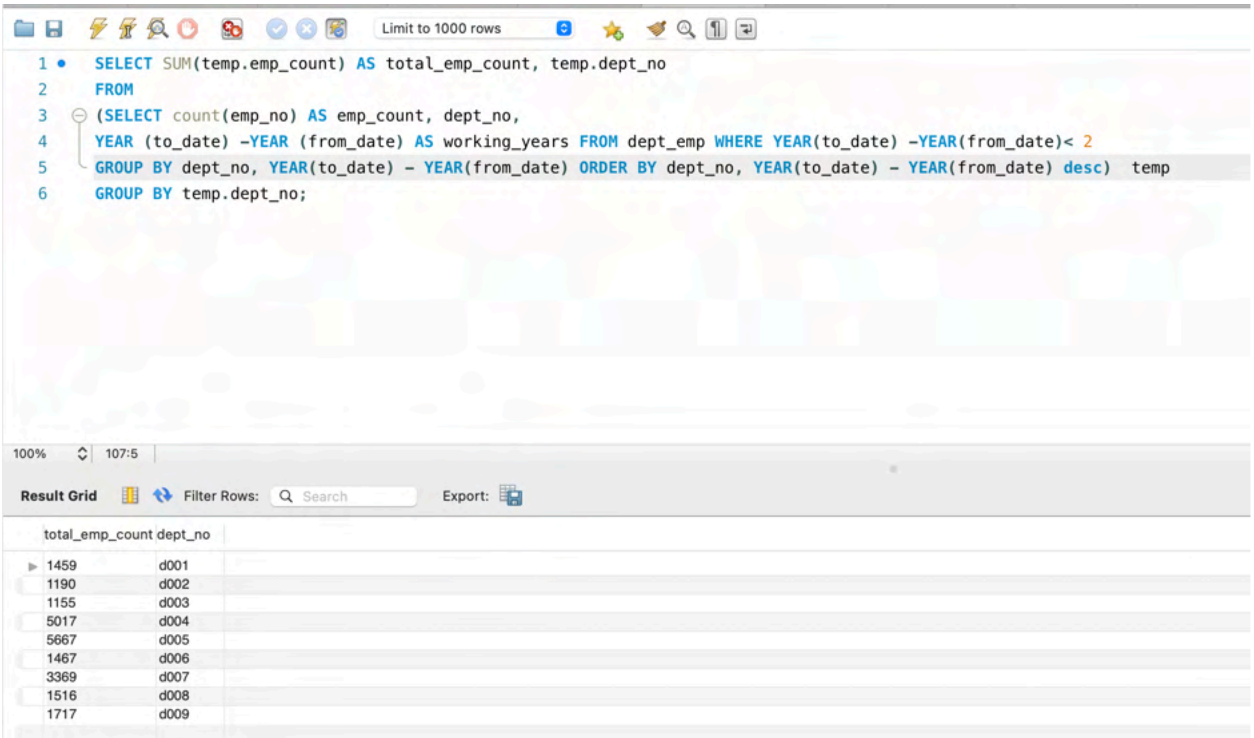
IV. Employees who are making more than the average salary in their department



```sql
select s1.emp_no, s1.salary, a.dept_no, a.avg_salary from salaries s1
inner join dept_emp de1 on de1.emp_no=s1.emp_no
Left join (select de.dept_no, avg(s.salary) as avg_salary from salaries s
inner join dept_emp de on de.emp_no=s.emp_no
group by de.dept_no) a on a.dept_no=de1.dept_no
where s1.salary > a.avg_salary
order by de1.dept_no desc;
```

emp_no	salary	dept_no	avg_salary
499965	59375	d009	58770.3665
499965	61773	d009	58770.3665
499965	62602	d009	58770.3665
499965	65009	d009	58770.3665
499965	67552	d009	58770.3665
499965	71926	d009	58770.3665
499965	75894	d009	58770.3665
499965	77577	d009	58770.3665
499965	79309	d009	58770.3665
499965	81318	d009	58770.3665
499965	83087	d009	58770.3665
499965	85645	d009	58770.3665
499965	89330	d009	58770.3665
499965	89714	d009	58770.3665
499965	90570	d009	58770.3665
499955	59338	d009	58770.3665
499955	59605	d009	58770.3665

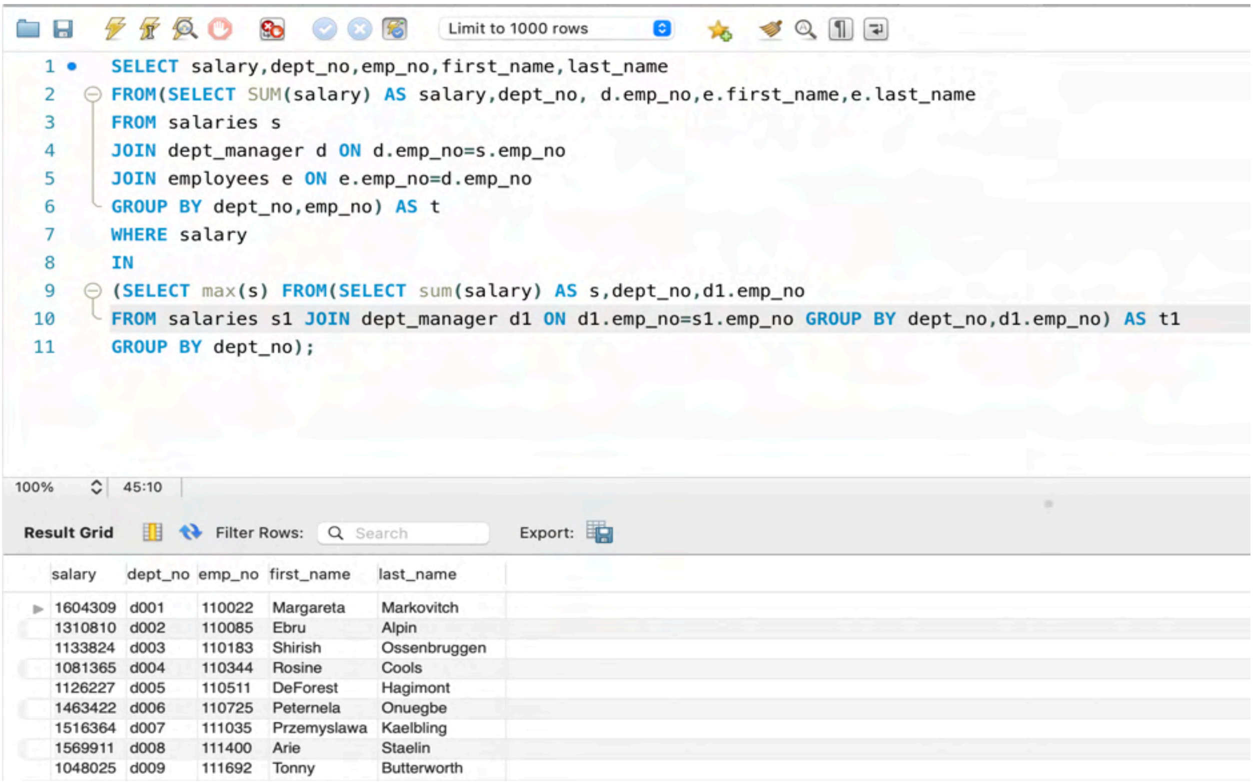
V. The number of employees who have been working for less than two years in each department.



```
SELECT SUM(temp.emp_count) AS total_emp_count, temp.dept_no
FROM
(SELECT count(emp_no) AS emp_count, dept_no,
YEAR (to_date) -YEAR (from_date) AS working_years FROM dept_emp WHERE YEAR(to_date) -YEAR(from_date)< 2
GROUP BY dept_no, YEAR(to_date) - YEAR(from_date) ORDER BY dept_no, YEAR(to_date) - YEAR(from_date) desc)  temp
GROUP BY temp.dept_no;
```

total_emp_count	dept_no
1459	d001
1190	d002
1155	d003
5017	d004
5667	d005
1467	d006
3369	d007
1516	d008
1717	d009

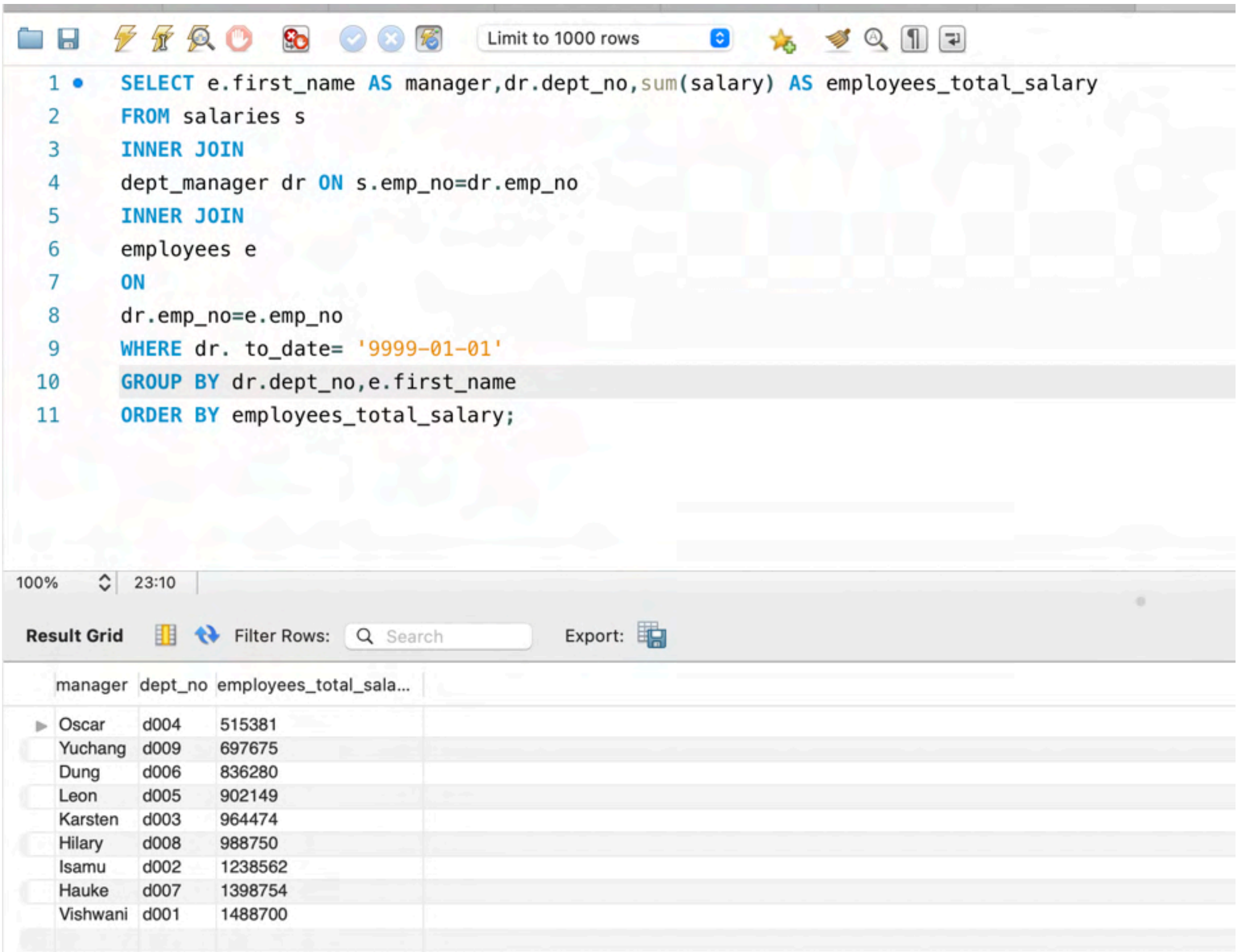
VI. Names of the employees (if any) who are drawing more than their managers.



```
SELECT salary,dept_no,emp_no,first_name,last_name
FROM(SELECT SUM(salary) AS salary,dept_no, d.emp_no,e.first_name,e.last_name
FROM salaries s
JOIN dept_manager d ON d.emp_no=s.emp_no
JOIN employees e ON e.emp_no=d.emp_no
GROUP BY dept_no,emp_no) AS t
WHERE salary
IN
(SELECT max(s) FROM(SELECT sum(salary) AS s,dept_no,d1.emp_no
FROM salaries s1 JOIN dept_manager d1 ON d1.emp_no=s1.emp_no GROUP BY dept_no,d1.emp_no) AS t1
GROUP BY dept_no);
```

salary	dept_no	emp_no	first_name	last_name
1604309	d001	110022	Margareta	Markovitch
1310810	d002	110085	Ebru	Alpin
1133824	d003	110183	Shirish	Ossenbruggen
1081365	d004	110344	Rosine	Cools
1126227	d005	110511	DeForest	Hagimont
1463422	d006	110725	Peternela	Onuegbe
1516364	d007	111035	Przemyslawa	Kaelbling
1569911	d008	111400	Arie	Staelin
1048025	d009	111692	Tonny	Butterworth

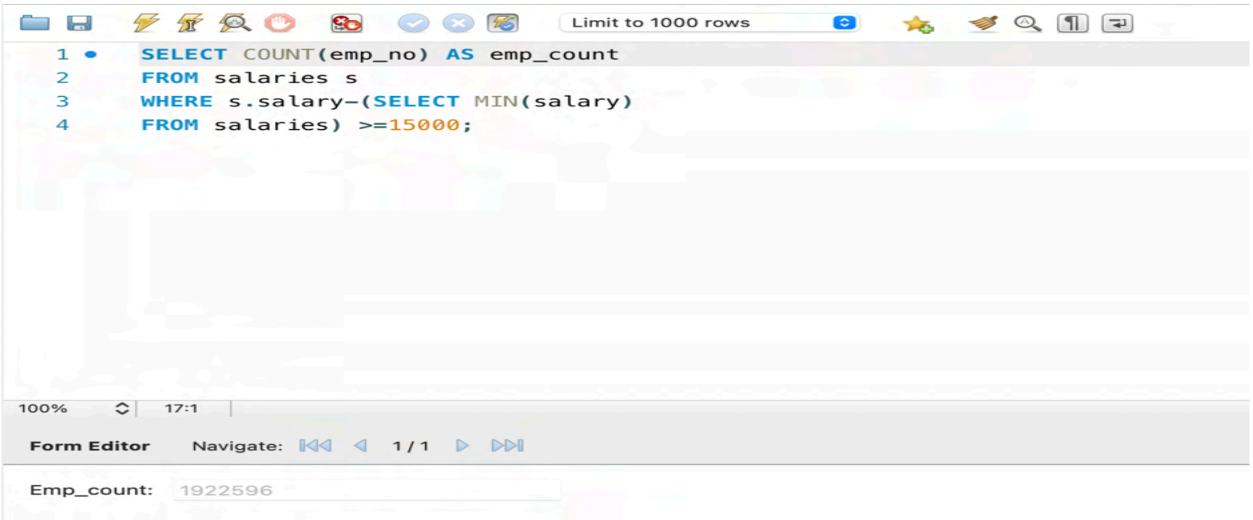
VII. Find the name of the manager of the department which pays the most salaries to its employees (total salary for all employees in a department)



```sql
SELECT e.first_name AS manager,dr.dept_no,sum(salary) AS employees_total_salary
FROM salaries s
INNER JOIN
dept_manager dr ON s.emp_no=dr.emp_no
INNER JOIN
employees e
ON
dr.emp_no=e.emp_no
WHERE dr. to_date= '9999-01-01'
GROUP BY dr.dept_no,e.first_name
ORDER BY employees_total_salary;
```

manager	dept_no	employees_total_sala...
Oscar	d004	515381
Yuchang	d009	697675
Dung	d006	836280
Leon	d005	902149
Karsten	d003	964474
Hilary	d008	988750
Isamu	d002	1238562
Hauke	d007	1398754
Vishwani	d001	1488700

VIII. Find the number of employees whose salary is at least 15,000 more than the least salary in the company



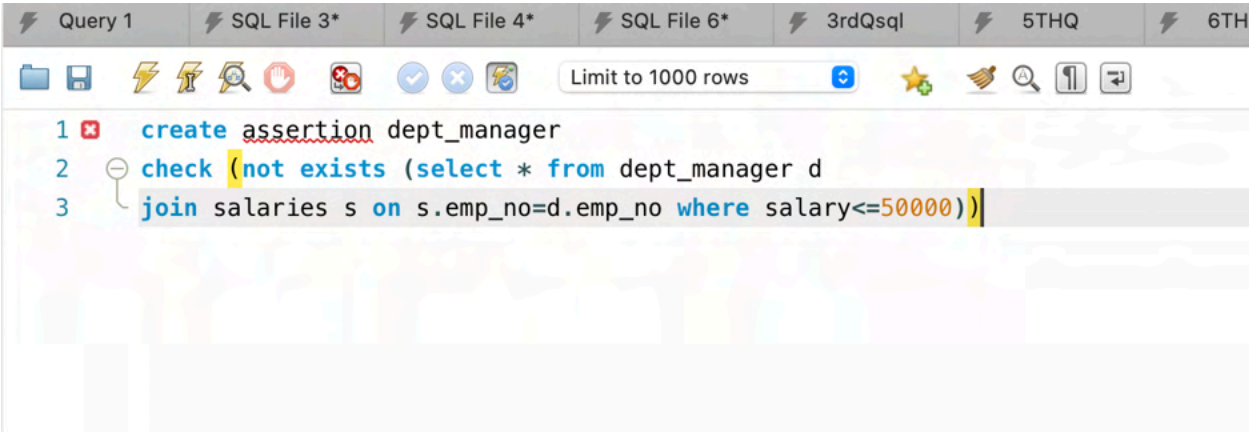
```sql
SELECT COUNT(emp_no) AS emp_count
FROM salaries s
WHERE s.salary-(SELECT MIN(salary)
FROM salaries) >=15000;
```

Emp_count: 1922596

Nikhil Mylarusetty- 016656393

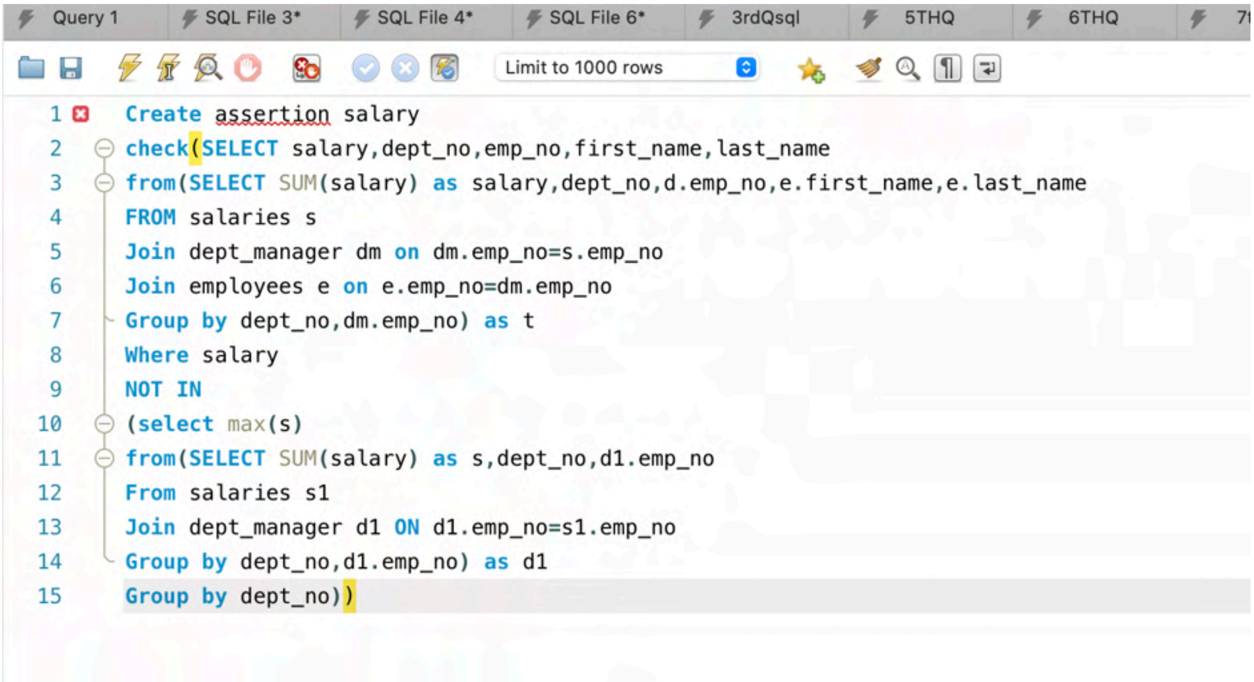
IX. Using the syntax discussed in class (SQL-92

reference: http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt Links to an external site.), write an assertion (may not execute) on dept_manager table which will ensure that all managers have a salary > 50,000



```
create assertion dept_manager
check (not exists (select * from dept_manager d
join salaries s on s.emp_no=d.emp_no where salary<=50000))
```

X. Similarly, write an assertion (may not execute) on salaries table which will ensure that no employee has a salary that is greater than his or her manager's salary



```
Create assertion salary
check(SELECT salary,dept_no,emp_no,first_name,last_name
from(SELECT SUM(salary) as salary,dept_no,d.emp_no,e.first_name,e.last_name
FROM salaries s
Join dept_manager dm on dm.emp_no=s.emp_no
Join employees e on e.emp_no=dm.emp_no
Group by dept_no,dm.emp_no) as t
Where salary
NOT IN
(select max(s)
from(SELECT SUM(salary) as s,dept_no,d1.emp_no
From salaries s1
Join dept_manager d1 ON d1.emp_no=s1.emp_no
Group by dept_no,d1.emp_no) as d1
Group by dept_no))
```

[10 points]

4. Explain one or more use case for each of the following joins. If it does not make sense or is invalid, state the same. Assume t1, t2, to be names of valid tables in an RDBMS and j,k,l,m,n their attributes (ignore syntax).

I. SELECT * FROM t1, t2;

- The above statement returns the cross join/ Cartesian product of tables t1 and t2 with all its attribute values.
- This syntax used cross join which gives the possibilities by combining all the records.
- For example, if table t1 have 4 records and t2 have 3 records then its cartesian product is 12 records, which is the resulting table.
- We can use cross joins when testing needs many rows.
- Cross joining or Cartesian products allow us to do things that inner joining cannot.

II. SELECT * FROM t1 a, t1 b;

- The above statement returns the cross join/ Cartesian product of all records (tables t1 & t2) with all its attribute values and using the aliases "a" and "b", the columns from t1 and t2 tables can also be accessed.
- Even this is same as the above syntax, where the cross join is used. So, if table 1 have 4 records and table2 have 4 records then the Cartesian product will be 16 records.

III. SELECT * FROM t1 a, t1 b WHERE a.k = b.m;

- The above query is representing the Cartesian product and inner join with a specified syntax.
- In table1 all the columns can access the alias name ‘a’ and table2 can access the alias name ‘b’.
- With the where clause, only the matching attributes from that particular column are returned as output; otherwise, the cartesian product is returned
- This query also represents inner join, which helps us to retrieve the data. Inner joins make sure that whichever tables have matching values will be picked.

IV. SELECT * FROM t1 a RIGHT JOIN t1 b USING (k);

- This query represents self-join and the type of join used is right join, so this focuses on the right table irrespective of the condition. Here, all the records from right table with alias name ‘b’ are fetched. In this case(self-join), we will have two sets of each attribute (J,L,M,N) from the table except the attribute (K) which is mentioned in the "USING" keyword.
- The joining operations are performed with the keywords 'ON' and 'USING'. In this query we used the 'USING' keyword (We use this keyword when there is same name in both the tables).
- We can use right joins when we wanted to focus on right table(all the rows) and match the values(rows) of the left table.

V. SELECT * FROM t1 RIGHT JOIN t2 ON t1.k = t2.k UNION ALL SELECT * FROM t1 LEFT JOIN t2 ON t1.k = t2.k

- In this query two joins are used, which are right join and left join. We also have Union All function, as the name implies, Union All is a procedure that combines the output of multiple tables into one table, in which the number of attributes (columns) and the data type of attributes from both queries should match and must be sorted in order. Two entries will be returned for each attribute (j,l,m,n) in the result. The first query is a right-join, and the second query is a left-join and self-join, so the second query result will also contain two entries, so the resultant query fetches all the records.

  [10 points]

5. Assume our RDBMS has just 3 tables:

Vendors(vid: integer, vname: string, speciality: string)

Equipment(eid: integer, ename: string, category: string)

PriceList(vid: integer, eid: integer, price: real)

Explain what each of the following 5 queries does, providing English interpretation of each clause. Ignore any typos and syntax issues.

i. SELECT E.ename FROM Equipment E, PriceList C, Vendors V

WHERE E.eid = C.eid AND C.vid = V.vid AND V.vname = 'ABC'

AND NOT EXISTS (SELECT * FROM PriceList C1, Vendors V1

WHERE E.eid = C1.eid AND C1.vid = V1.vid AND

V1.vname <> 'ABC')

- The outer query returns Equipment name (E.ename) sold by the vendor with name 'ABC'. So basically 3 tables are joined here, i.e., equipment and pricelist with E.eid = C.eid, pricelist and vendor table with C.vid = V.vid. And the condition to get the records of a particular vendor is V.vname = 'ABC'.
- And the sub query with 'NOT EXIST' selects all the records of pricelist and vendors tables which satisfy the conditions E.eid = C1.eid and C1.vid = V1.vid, and the condition vendors name not equal to the vendor name 'ABC'.
- So, when there is a vendor with name 'ABC', the outer query returns true and the 'NOT EXIST' query is set to be false.

Nikhil Mylarusetty- 016656393

ii. SELECT V.vname, COUNT(*) as ToolsCount

FROM Vendors V, Equipment E, PriceList C WHERE E.eid = C.eid AND C.vid = V.vid

GROUP BY V.vname, V.vid HAVING EVERY (E.category=’Electronics’)

- This query returns the names of the vendors and the tool count of their equipment in electronic category only.
- So basically here we are selecting vendor name and counting all the rows as tool count from Vendors V, Equipment E, Pricelist C using the conditions E.eid = C.eid and C.vid = V.vid and grouping them with vendor name and vendor Id. Using the having clause, grouped records are filtered according to the category in the equipment table equal to electronics.

iii. SELECT DISTINCT C.vid FROM PriceList C, Equipment E

WHERE C.eid = E.eid AND E.category = ‘Mechanical’

UNION

SELECT DISTINCT C1.vid FROM PriceList C1, Equipment E1

WHERE C1.eid = E1.eid AND E1.category = ‘Electronics’

- Several vendor IDs in the pricelist are returned from the query, and they are all sellers of equipment of the “Mechanical” and “Electronic” categories
- We can see from the syntax that this is a union of two queries. So, When two queries are joined, union will combine the results and also eliminate duplicates from the resulting table.
- From the two queries, in the first query, with ID as primary key in one table and foreign key in another table, the query provides the distinct number of vendor IDs in pricelist and equipment tables where the equipment category is mechanical.
- And from equipment and pricelist table using primary keys and foreign keys, the second query will return the number of distinct vendor IDS for the 'Electronics' equipment category.
- And then Union function will combine all the records with common distinct vendor ID’s.

```
iv. SELECT E.eid, V.vname FROM Equipment E, Vendors V, PriceList C

WHERE C.eid = E.eid AND C.vid = V.vid

AND C.price = (SELECT MAX (C1.price)

FROM PriceList C1 WHERE C1.eid = E.eid)
```

- In the above query, we retrieve the equipment ID and vendor name of the equipment whose price is the highest among all the equipment's prices
- Inner query retrieve the most expensive equipment from the table pricelist alias C1.
- Using the joining conditions primary key = foreign key and price = maximum price returned from the sub query, the outer query selects equipment_Id and vendor name from the resulting table.

```
v. SELECT DISTINCT C.vid FROM PriceList C

WHERE C.price > ( SELECT AVG (C1.price)

                          FROM PriceList C1 WHERE C1.eid = C.eid )
```

- This query displays the vendor IDs whose prices are greater than average from the priceList table.