Nhat Nguyen

Assignment 3 – CSCI 4140

Contents

Tools used: ... 3

Create database ... 4

Create an application and connect to the database ... 10

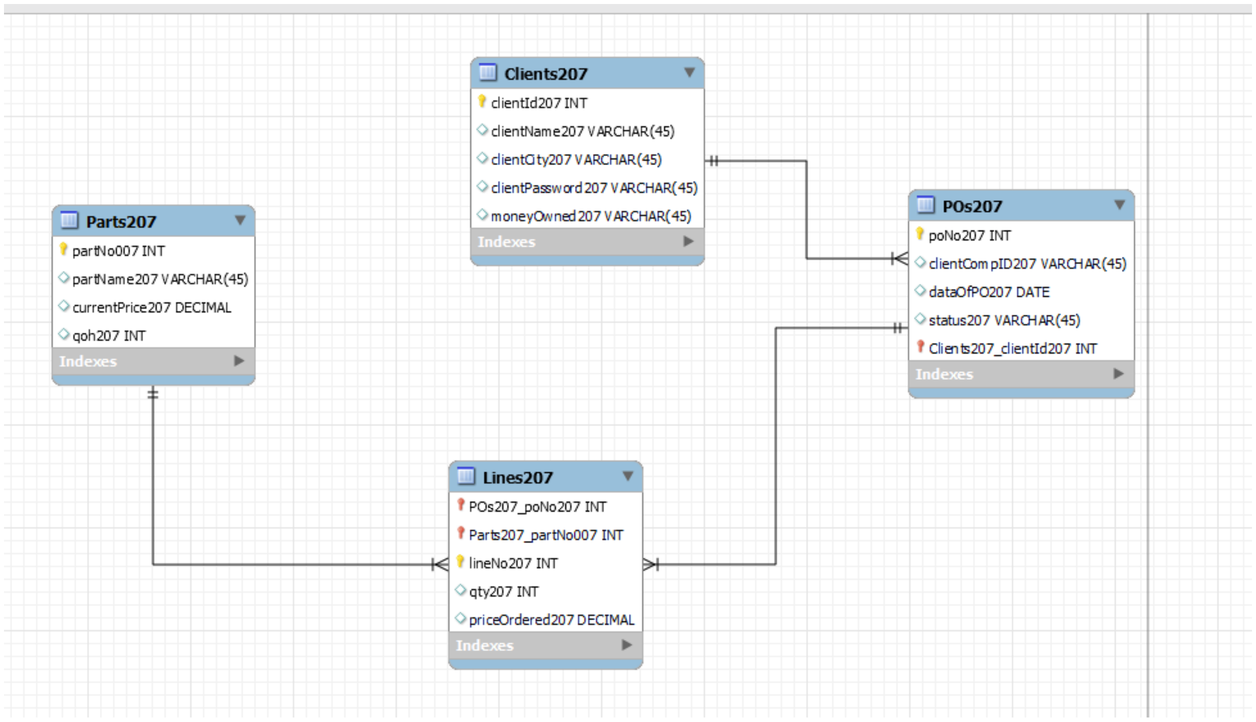
Methods to query database ... 14

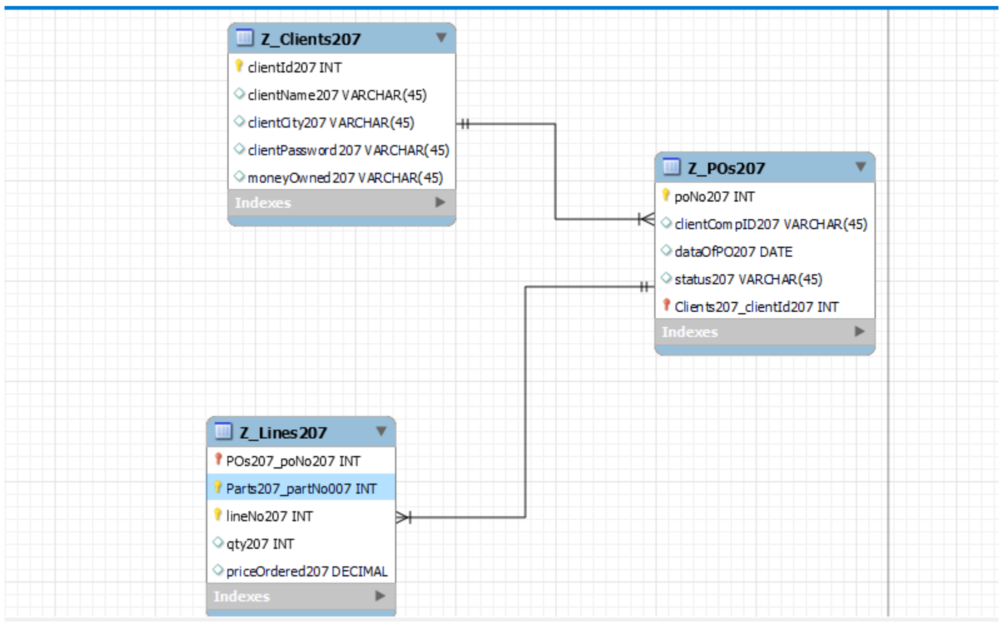
Tools used and Git repo

- MySQL Work Bench Community was used to run a localhost server
- Bootstrap was used for UI
- Since I have little experience with backend on Java, I decided to use Python Flask framework for backend
- Git repository link: https://github.com/nhatnguyen215/A3_CSCI4140

Create database

- Similar to assignment 1, database was created using forward engineering after the models are created. For this one though, I created 3 databases, the first 2 are similar to the database in assignment 1 and 2, for the third one for company Z, we just need to remove Parts table



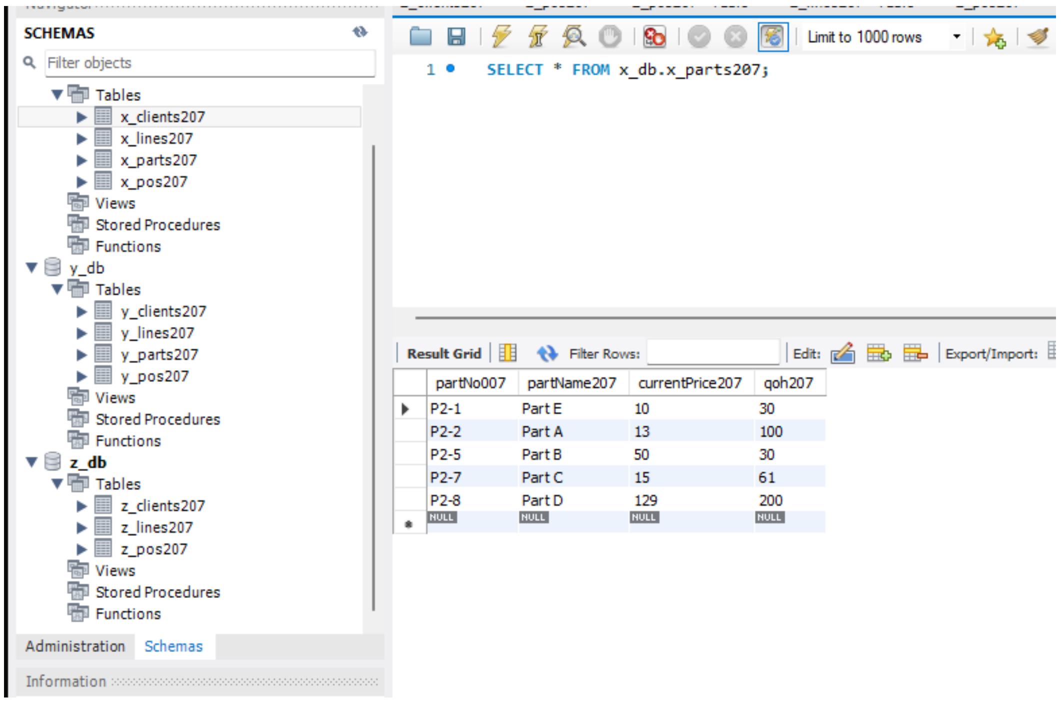


- Below is the code generated from the EER diagram for database Z, the first 2 databases are similar to the first and second assignments, so I thought I won't have to include it here:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DA
TE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema Z_db
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema Z_db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `Z_db` DEFAULT CHARACTER SET utf8 ;
USE `Z_db` ;

-- -----------------------------------------------------
-- Table `Z_db`.`Z_Clients207`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Z_db`.`Z_Clients207` (
  `clientId207` INT NOT NULL,
  `clientName207` VARCHAR(45) NULL,
  `clientCity207` VARCHAR(45) NULL,
  `clientPassword207` VARCHAR(45) NULL,
  `moneyOwned207` VARCHAR(45) NULL,
  PRIMARY KEY (`clientId207`),
  UNIQUE INDEX `clientId207_UNIQUE` (`clientId207` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Z_db`.`Z_POs207`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Z_db`.`Z_POs207` (
  `poNo207` INT NOT NULL,
  `clientCompID207` VARCHAR(45) NULL,
  `dataOfPO207` DATE NULL,
  `status207` VARCHAR(45) NULL,
  `Clients207_clientId207` INT NOT NULL,
  PRIMARY KEY (`poNo207`, `Clients207_clientId207`),
  UNIQUE INDEX `poNo207_UNIQUE` (`poNo207` ASC) VISIBLE,
  INDEX `fk_POs207_Clients2071_idx` (`Clients207_clientId207` ASC) VISIBLE,
  CONSTRAINT `fk_POs207_Clients2071`
    FOREIGN KEY (`Clients207_clientId207`)
    REFERENCES `Z_db`.`Z_Clients207` (`clientId207`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Z_db`.`Z_Lines207`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Z_db`.`Z_Lines207` (
  `POs207_poNo207` INT NOT NULL,
  `Parts207_partNo007` INT NOT NULL,
  `lineNo207` INT NOT NULL,
  `qty207` INT NULL,
  `priceOrdered207` DECIMAL NULL,
  PRIMARY KEY (`POs207_poNo207`, `Parts207_partNo007`, `lineNo207`),
  INDEX `fk_POs207_has_Parts207_POs207_idx` (`POs207_poNo207` ASC) VISIBLE,
  CONSTRAINT `fk_POs207_has_Parts207_POs207`
    FOREIGN KEY (`POs207_poNo207`)
    REFERENCES `Z_db`.`Z_POs207` (`poNo207`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

- Screenshots of table showed in MySQL Workbench with dump data:

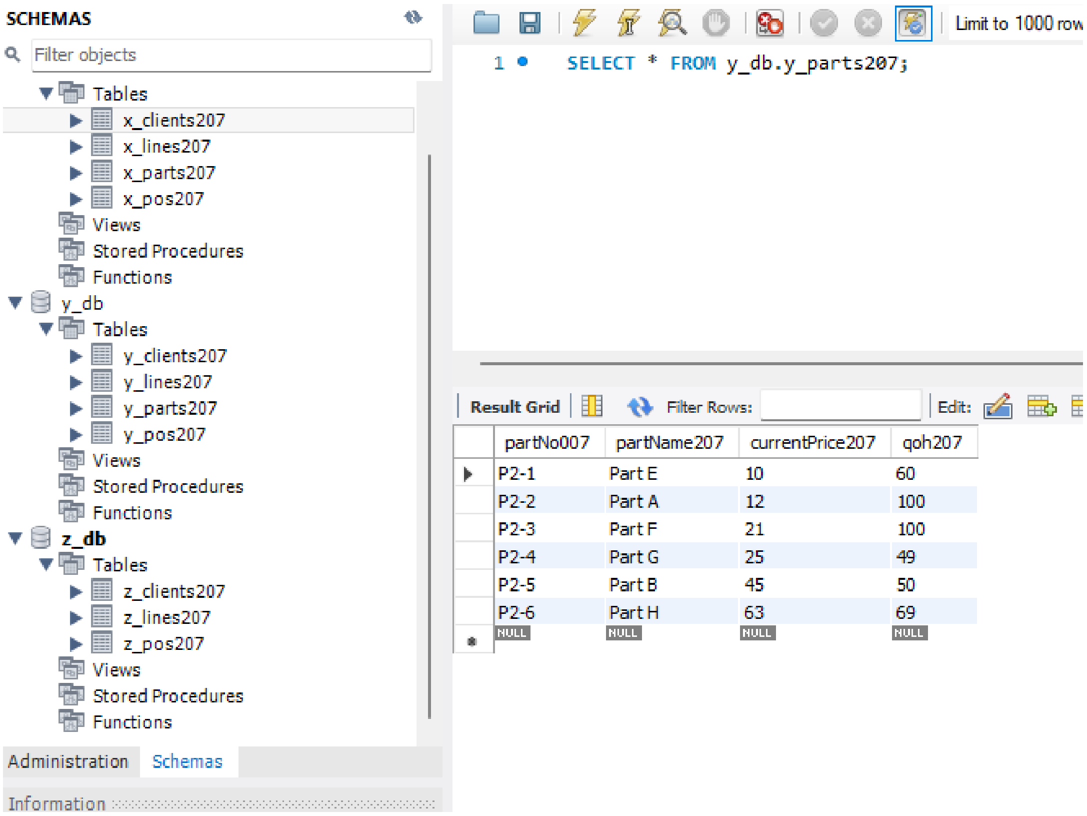
Company X parts:



```
SELECT * FROM x_db.x_parts207;
```

partNo007	partName207	currentPrice207	qoh207
P2-1	Part E	10	30
P2-2	Part A	13	100
P2-5	Part B	50	30
P2-7	Part C	15	61
P2-8	Part D	129	200
NULL	NULL	NULL	NULL

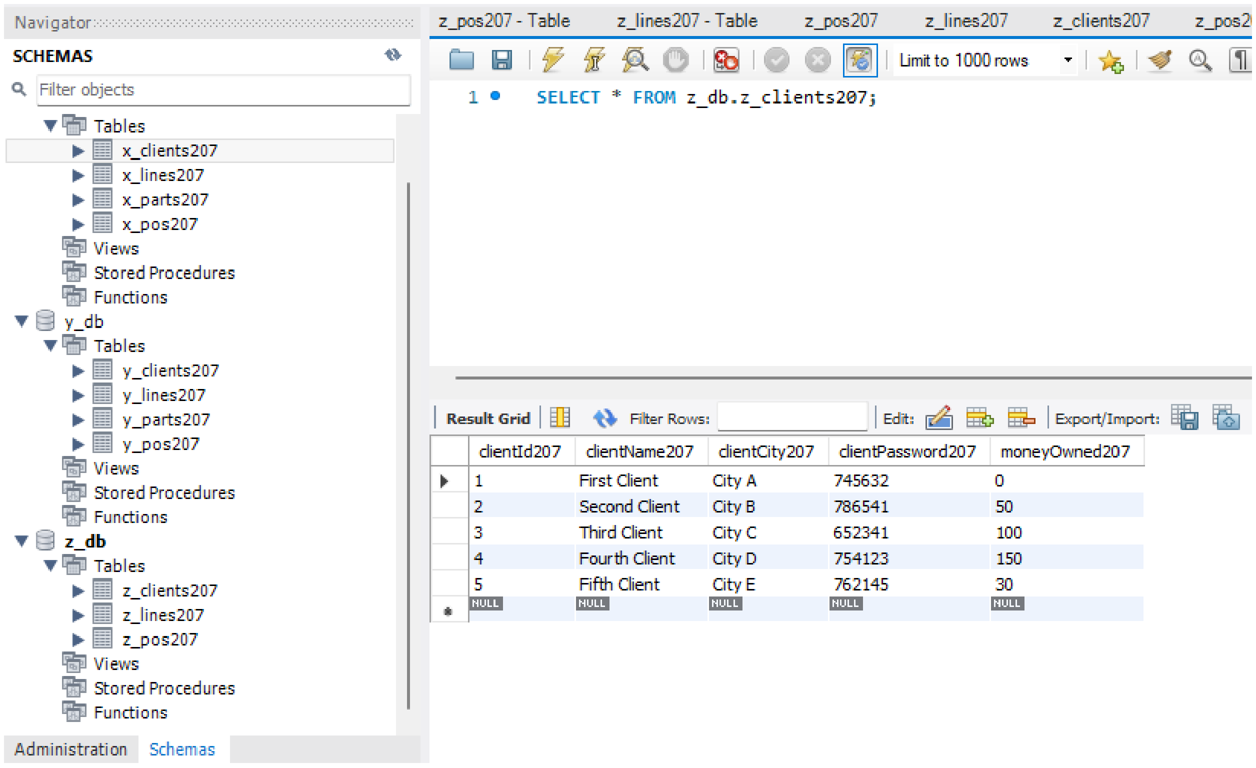
Company Y parts:



```
SELECT * FROM y_db.y_parts207;
```

partNo007	partName207	currentPrice207	qoh207
P2-1	Part E	10	60
P2-2	Part A	12	100
P2-3	Part F	21	100
P2-4	Part G	25	49
P2-5	Part B	45	50
P2-6	Part H	63	69
NULL	NULL	NULL	NULL

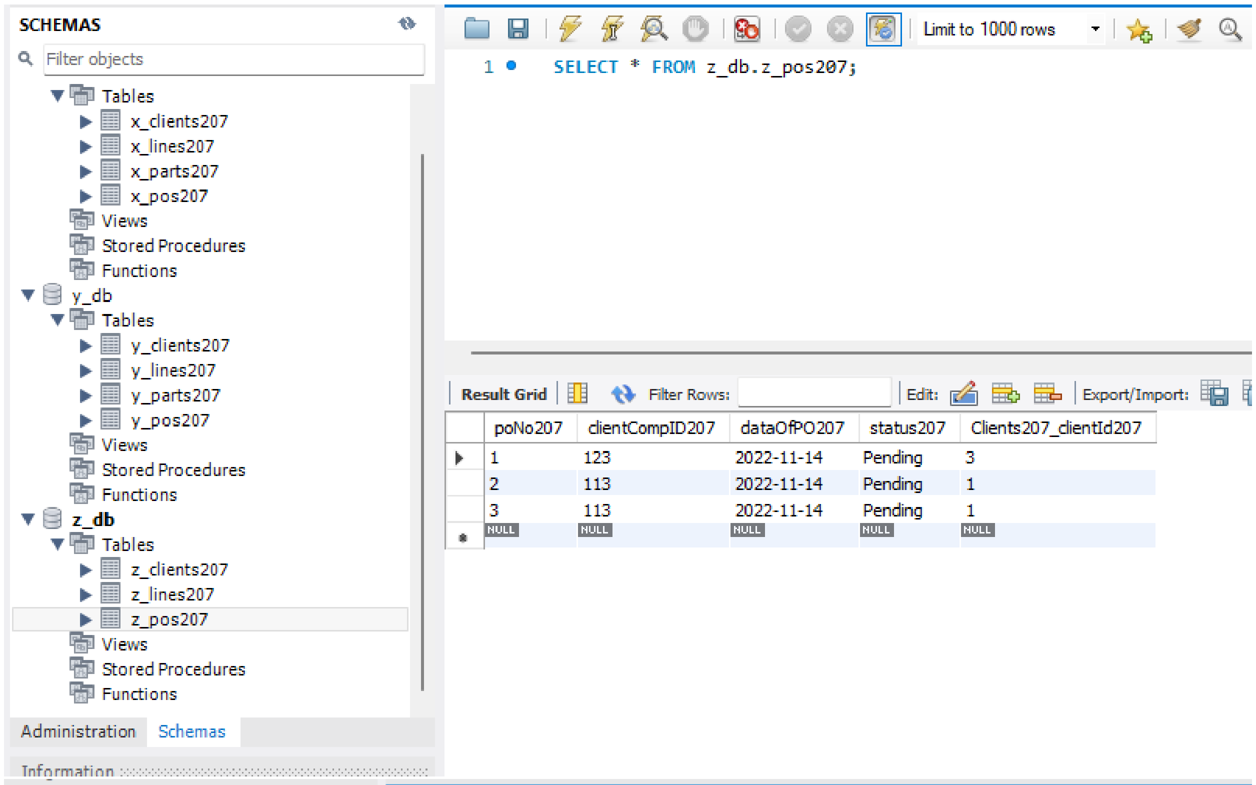
Clients in company Z:



```
SELECT * FROM z_db.z_clients207;
```

clientId207	clientName207	clientCity207	clientPassword207	moneyOwned207
1	First Client	City A	745632	0
2	Second Client	City B	786541	50
3	Third Client	City C	652341	100
4	Fourth Client	City D	754123	150
5	Fifth Client	City E	762145	30
NULL	NULL	NULL	NULL	NULL

Some other dump data in company Z:

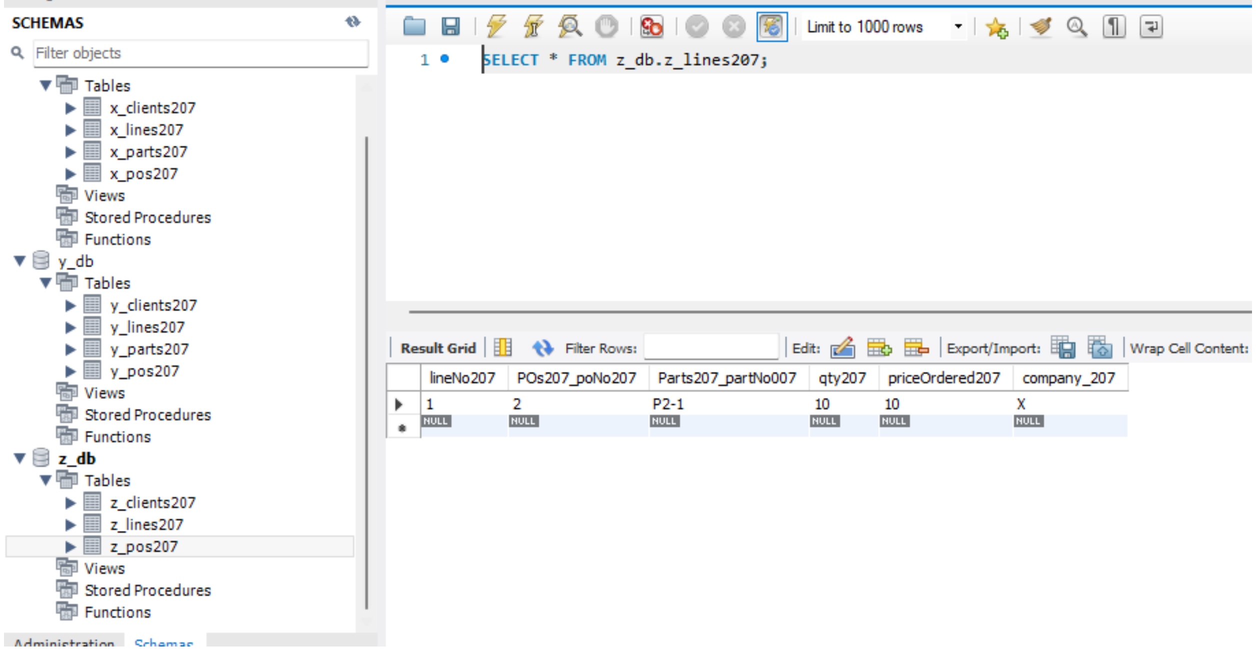


SCHEMAS

- Tables
  - x_clients207
  - x_lines207
  - x_parts207
  - x_pos207
- Views
- Stored Procedures
- Functions
- y_db
  - Tables
    - y_clients207
    - y_lines207
    - y_parts207
    - y_pos207
  - Views
  - Stored Procedures
  - Functions
- z_db
  - Tables
    - z_clients207
    - z_lines207
    - z_pos207
  - Views
  - Stored Procedures
  - Functions

```
SELECT * FROM z_db.z_pos207;
```

poNo207	clientCompID207	dataOfPO207	status207	Clients207_clientId207
1	123	2022-11-14	Pending	3
2	113	2022-11-14	Pending	1
3	113	2022-11-14	Pending	1
NULL	NULL	NULL	NULL	NULL



SCHEMAS

- Tables
  - x_clients207
  - x_lines207
  - x_parts207
  - x_pos207
- Views
- Stored Procedures
- Functions
- y_db
  - Tables
    - y_clients207
    - y_lines207
    - y_parts207
    - y_pos207
  - Views
  - Stored Procedures
  - Functions
- z_db
  - Tables
    - z_clients207
    - z_lines207
    - z_pos207
  - Views
  - Stored Procedures
  - Functions

```
SELECT * FROM z_db.z_lines207;
```

lineNo207	POs207_poNo207	Parts207_partNo007	qty207	priceOrdered207	company_207
1	2	P2-1	10	10	X
NULL	NULL	NULL	NULL	NULL	NULL

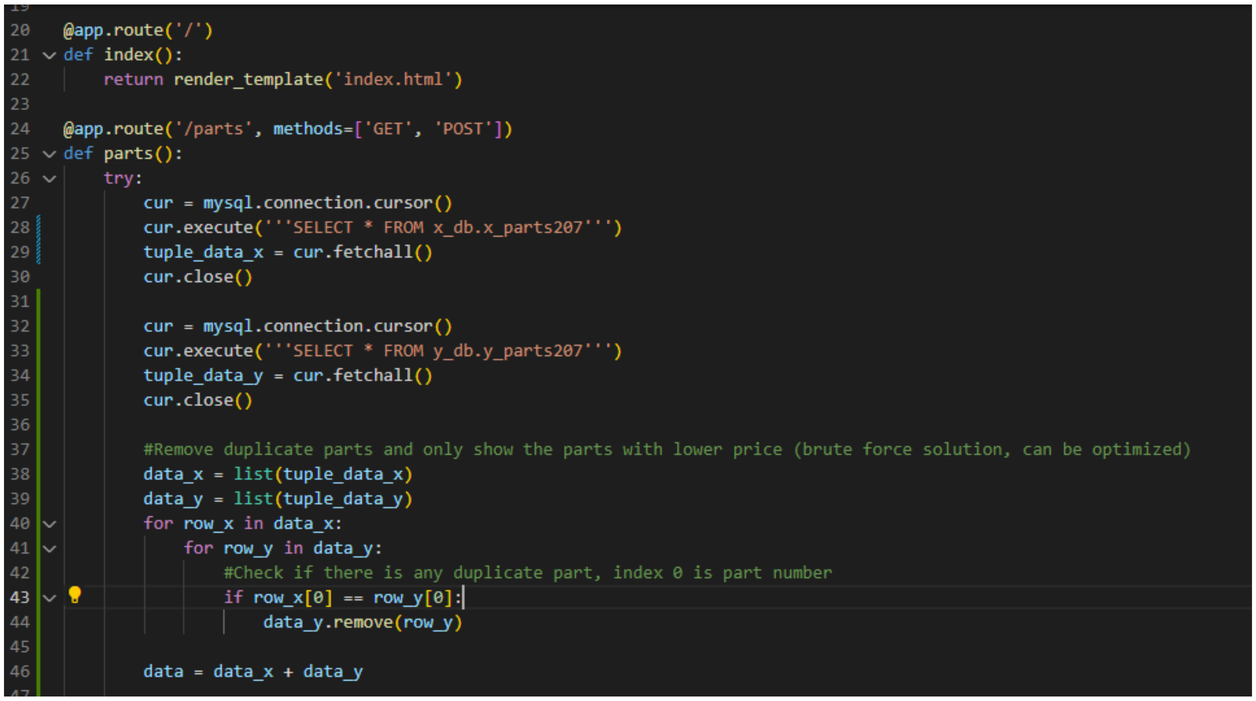
Create an application and connect to the database

The web application is run on a virtual environment using Python flask framework, the setup process can be found here: https://flask.palletsprojects.com/en/2.2.x/installation/

Firstly, I link the framework with the local database, with the following config:

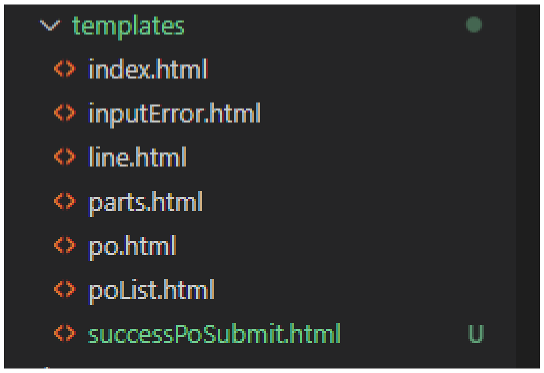
```python
from flask import Flask, render_template, request, flash
from flask_mysqldb import MySQL
import random, datetime

app = Flask(__name__)
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'admin'
app.config['MYSQL_DB'] = 'mydb'

mysql = MySQL(app)
```

The web app will have different routes (pages) for each method we will be running (Look up parts, submit PO, list line, list PO):

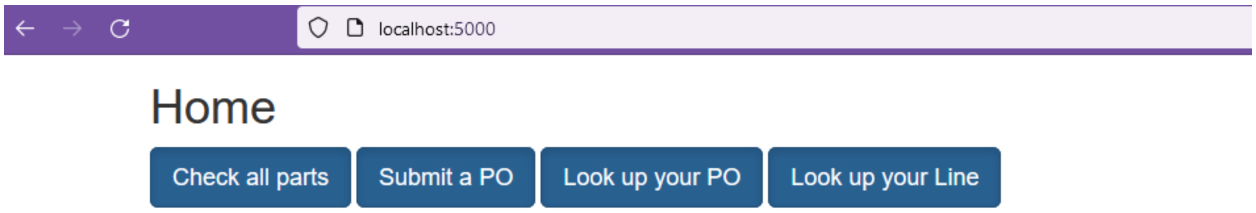


```python
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/parts', methods=['GET', 'POST'])
def parts():
    try:
        cur = mysql.connection.cursor()
        cur.execute('''SELECT * FROM x_db.x_parts207''')
        tuple_data_x = cur.fetchall()
        cur.close()

        cur = mysql.connection.cursor()
        cur.execute('''SELECT * FROM y_db.y_parts207''')
        tuple_data_y = cur.fetchall()
        cur.close()

        #Remove duplicate parts and only show the parts with lower price (brute force solution, can be optimized)
        data_x = list(tuple_data_x)
        data_y = list(tuple_data_y)
        for row_x in data_x:
            for row_y in data_y:
                #Check if there is any duplicate part, index 0 is part number
                if row_x[0] == row_y[0]:
                    data_y.remove(row_y)

        data = data_x + data_y
```

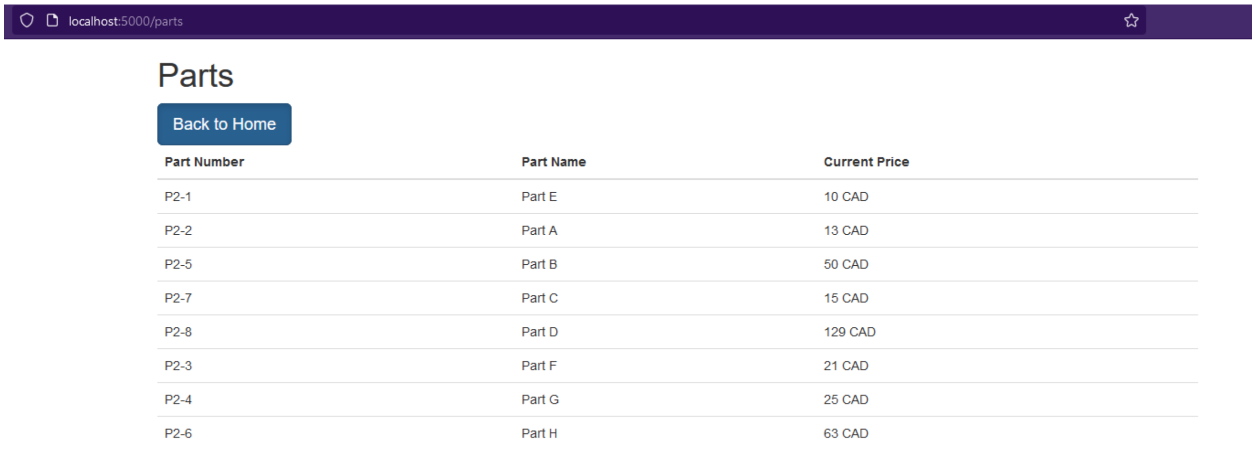
After that, I created html pages and link them to their route


```
templates
  index.html
  inputError.html
  line.html
  parts.html
  po.html
  poList.html
  successPoSubmit.html   U
```

Home page:



Parts page that list all the parts available, with all parts provided by company X and Z, no duplicates:

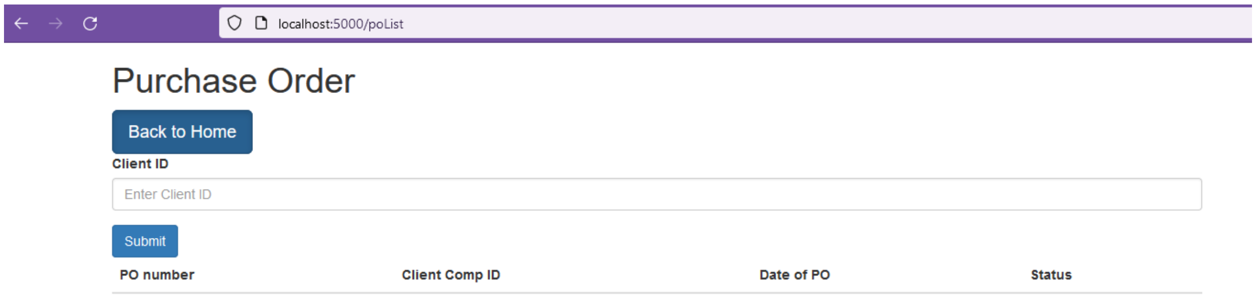


localhost:5000/parts

Parts

Part Number	Part Name	Current Price
P2-1	Part E	10 CAD
P2-2	Part A	13 CAD
P2-5	Part B	50 CAD
P2-7	Part C	15 CAD
P2-8	Part D	129 CAD
P2-3	Part F	21 CAD
P2-4	Part G	25 CAD
P2-6	Part H	63 CAD

PO page where you can find your POs connected to your client ID:

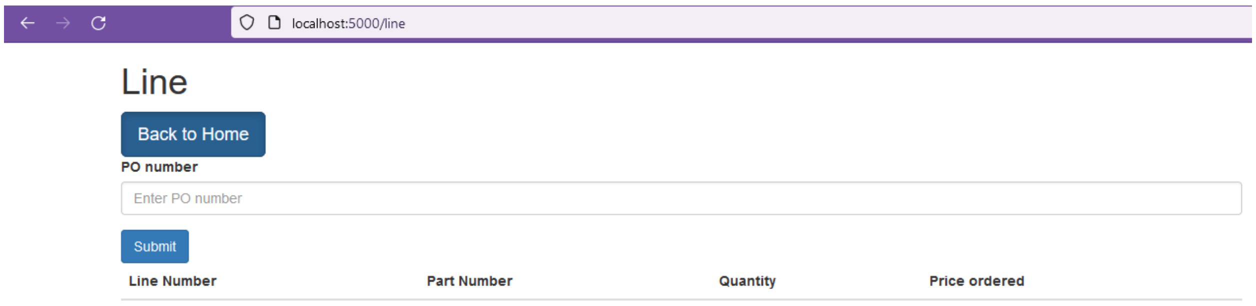


localhost:5000/poList

Purchase Order

PO number	Client Comp ID	Date of PO	Status

Line page where you can look up your line based on PO number:


localhost:5000/line

Line

Back to Home

PO number

Enter PO number

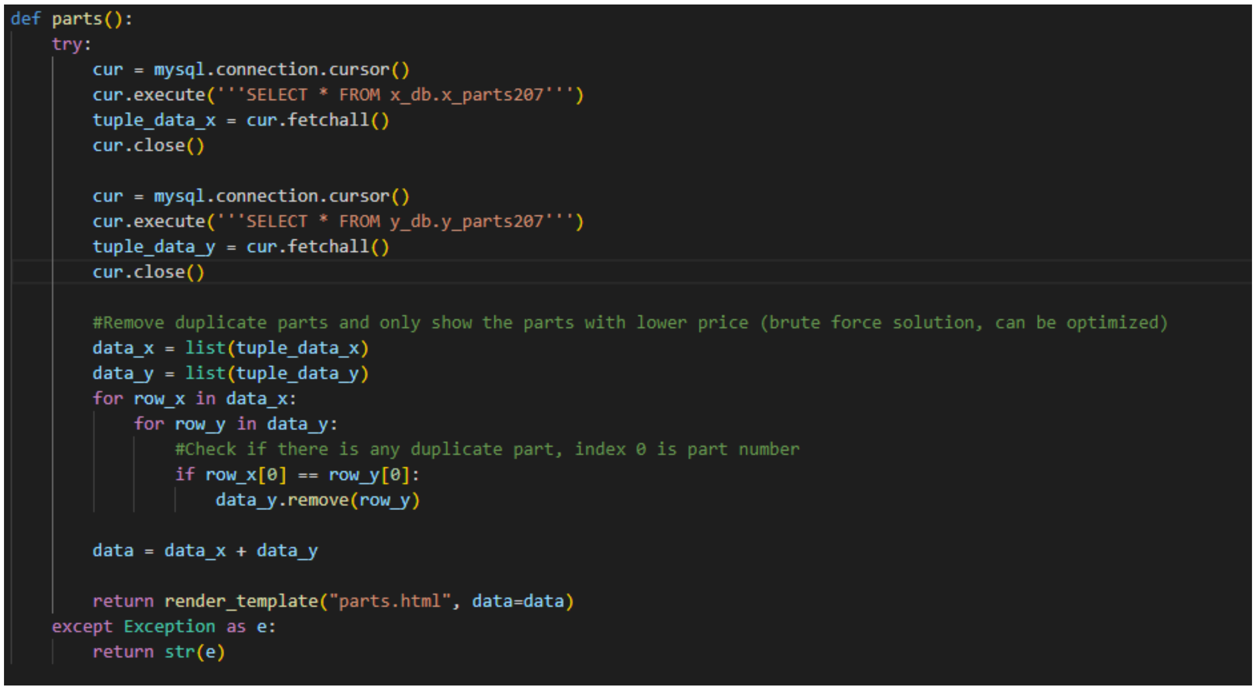
Submit

Line Number	Part Number	Quantity	Price ordered

Methods to query database

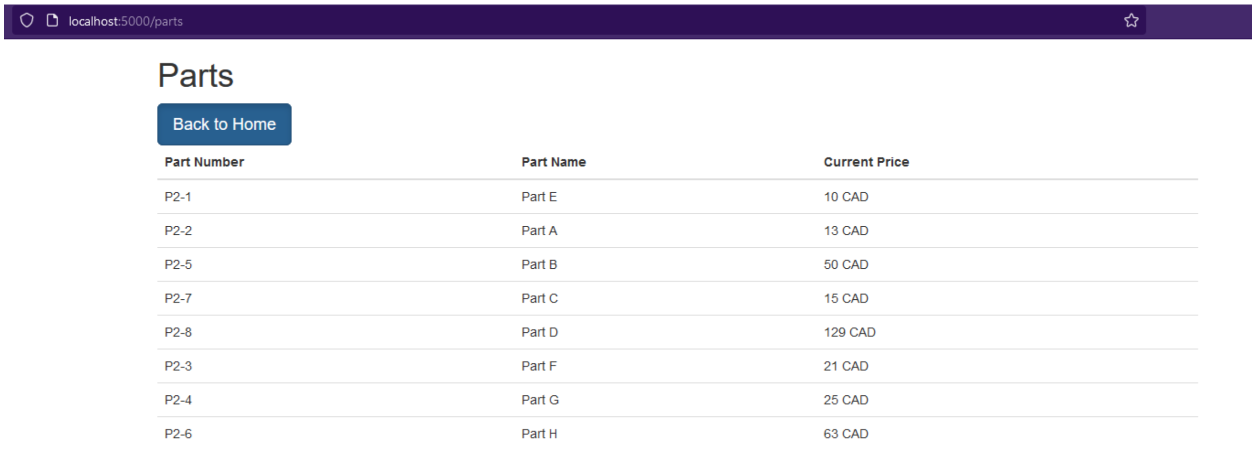
All the methods were written in run.py file:

- On Part page, we want to query all the parts available at company X and Y, but we have to remove all the duplicates, function parts() was created for that:



```python
def parts():
    try:
        cur = mysql.connection.cursor()
        cur.execute('''SELECT * FROM x_db.x_parts207''')
        tuple_data_x = cur.fetchall()
        cur.close()

        cur = mysql.connection.cursor()
        cur.execute('''SELECT * FROM y_db.y_parts207''')
        tuple_data_y = cur.fetchall()
        cur.close()

        #Remove duplicate parts and only show the parts with lower price (brute force solution, can be optimized)
        data_x = list(tuple_data_x)
        data_y = list(tuple_data_y)
        for row_x in data_x:
            for row_y in data_y:
                #Check if there is any duplicate part, index 0 is part number
                if row_x[0] == row_y[0]:
                    data_y.remove(row_y)

        data = data_x + data_y

        return render_template("parts.html", data=data)
    except Exception as e:
        return str(e)
```

- When click on find part on the home page, I used get method to return all the information of parts table from the database:

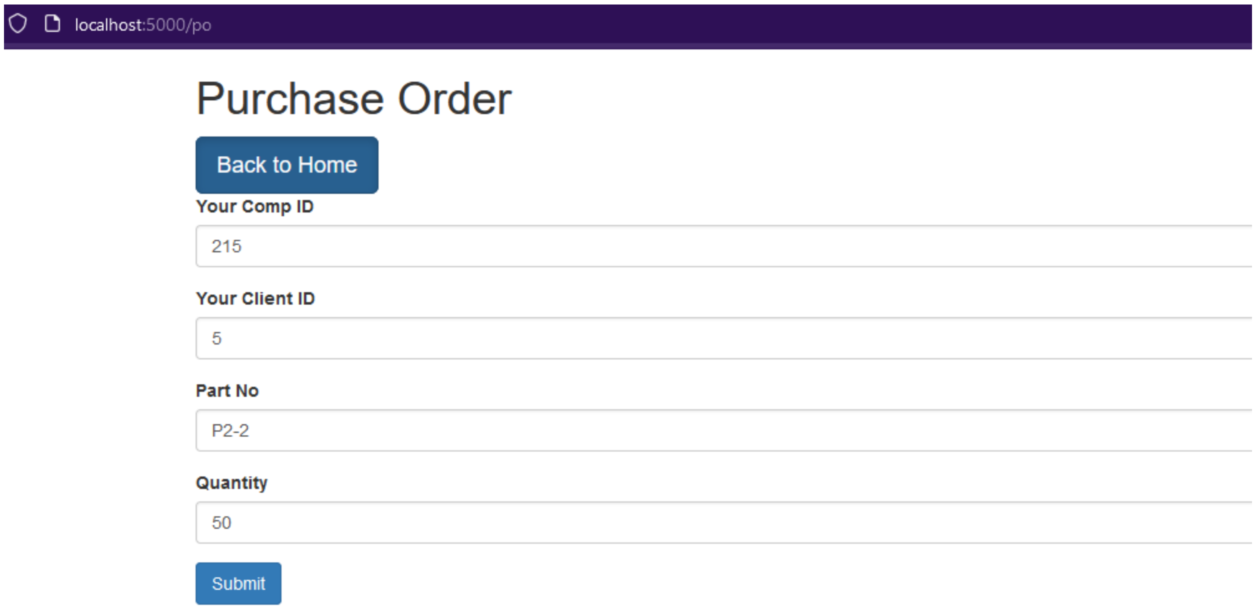


localhost:5000/parts

Parts

Part Number	Part Name	Current Price
P2-1	Part E	10 CAD
P2-2	Part A	13 CAD
P2-5	Part B	50 CAD
P2-7	Part C	15 CAD
P2-8	Part D	129 CAD
P2-3	Part F	21 CAD
P2-4	Part G	25 CAD
P2-6	Part H	63 CAD

- On PO page, we want to submit a purchase order to company Z, and also insert a new line that connects part to that order, with the correct company we want to make the PO from, first we want to get information from customer:



localhost:5000/po

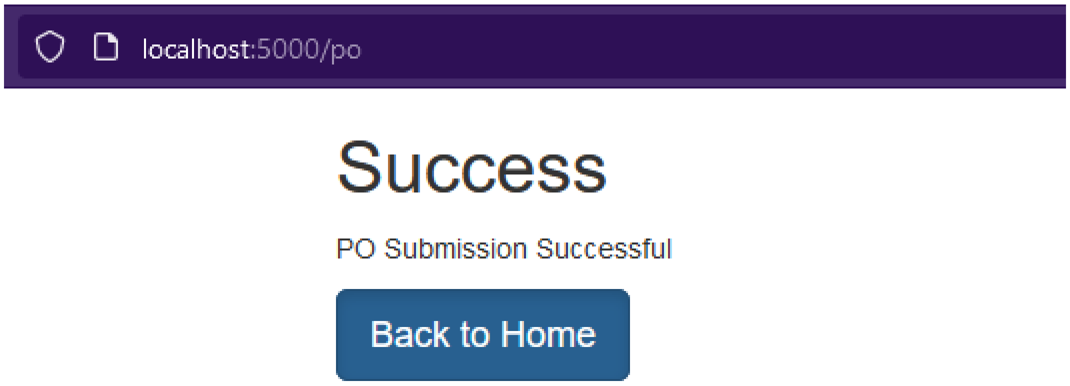
Purchase Order

Back to Home

Your Comp ID	215
Your Client ID	5
Part No	P2-2
Quantity	50

Submit

- Submission is successful, since P2-2 is provided by both companies X and Y, however Y has lower price, so the line should mention the PO is to make with company Y



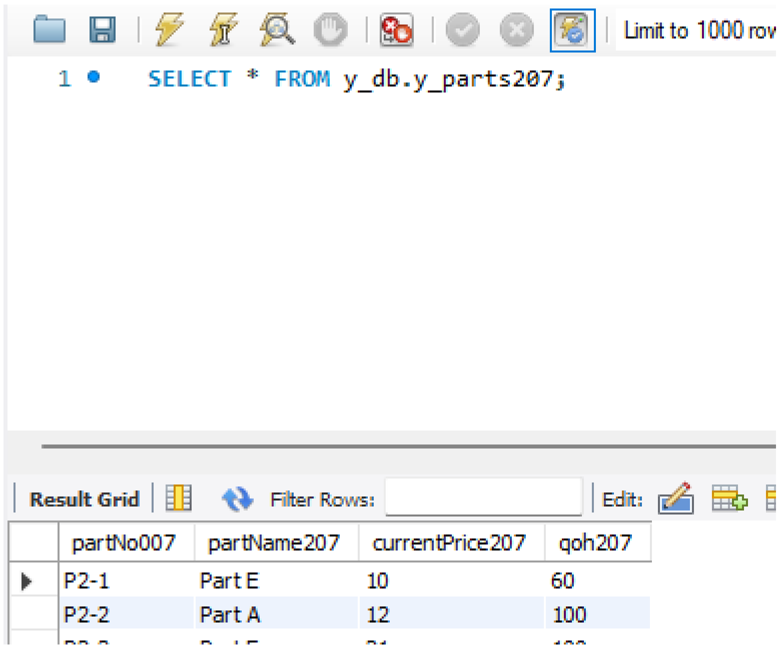
localhost:5000/po

Success

PO Submission Successful

Back to Home

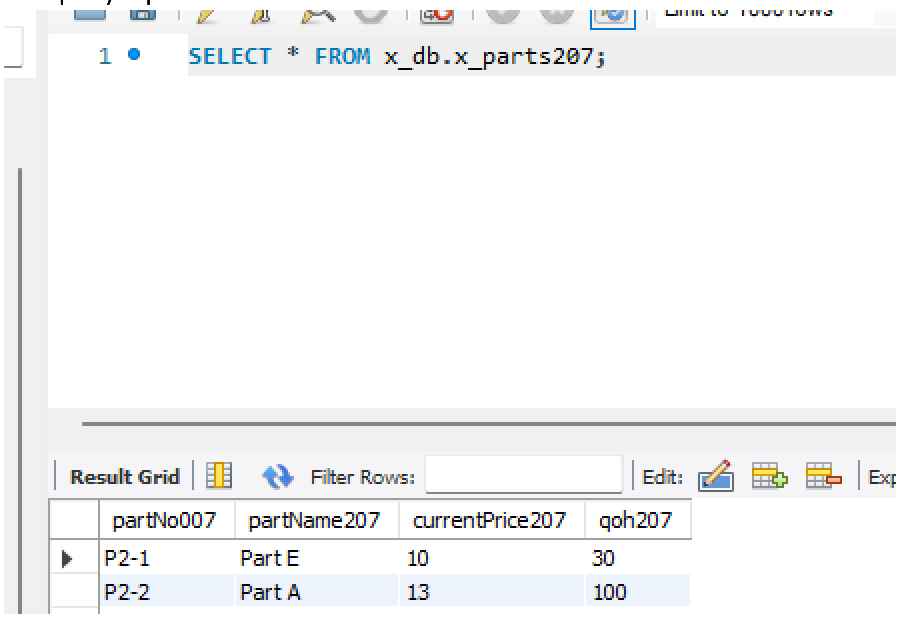
Company y price for P2-2 is 12:



```
SELECT * FROM y_db.y_parts207;
```

	partNo007	partName207	currentPrice207	qoh207
▶	P2-1	Part E	10	60
	P2-2	Part A	12	100

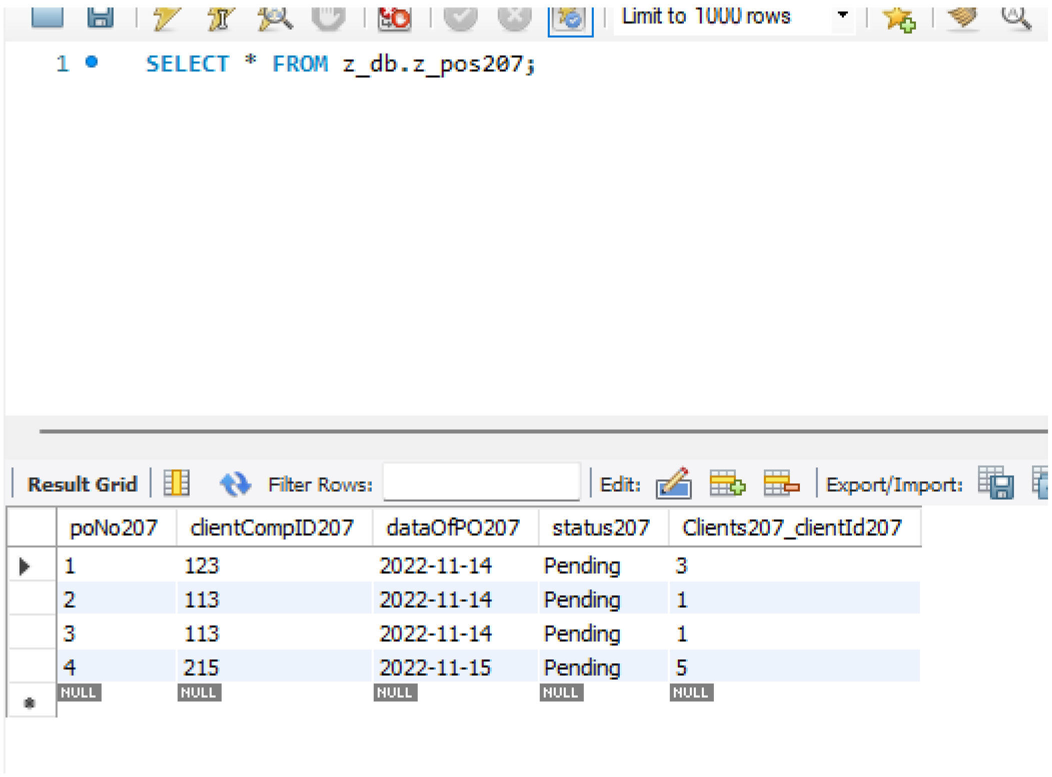
Company X price for P2-2 is 13:



```
SELECT * FROM x_db.x_parts207;
```

	partNo007	partName207	currentPrice207	qoh207
▶	P2-1	Part E	10	30
	P2-2	Part A	13	100

- After submitting the form, database has been updated with new PO and line:

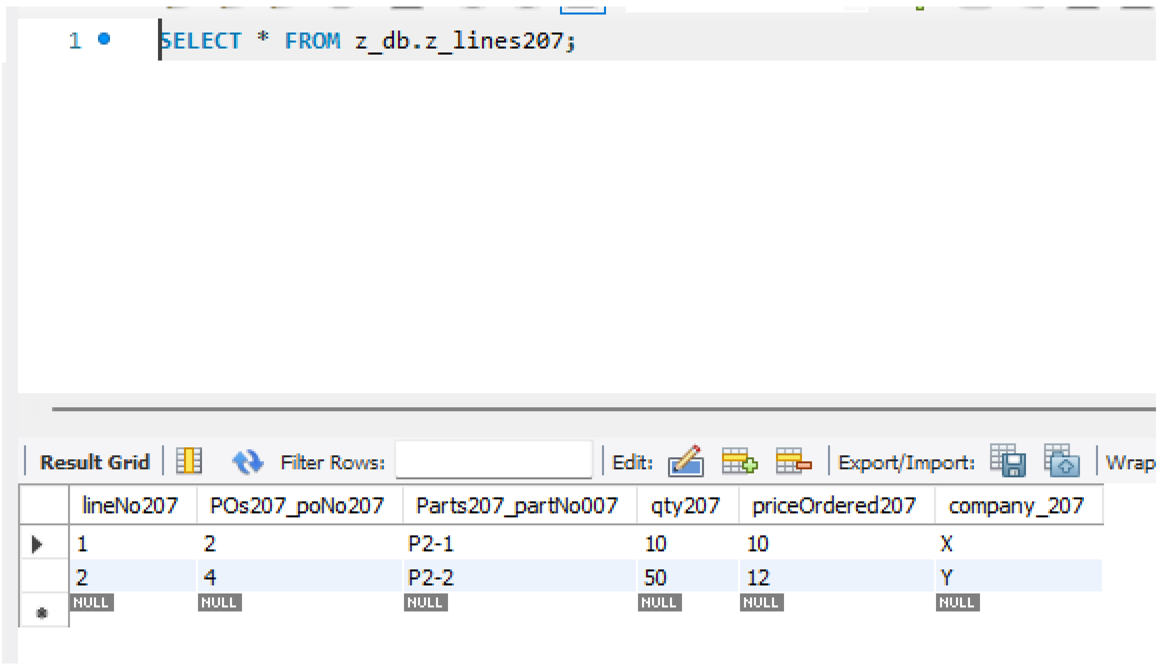


Limit to 1000 rows

```
SELECT * FROM z_db.z_pos207;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

poNo207	clientCompID207	dataOfPO207	status207	Clients207_clientId207
1	123	2022-11-14	Pending	3
2	113	2022-11-14	Pending	1
3	113	2022-11-14	Pending	1
4	215	2022-11-15	Pending	5
NULL	NULL	NULL	NULL	NULL



```
SELECT * FROM z_db.z_lines207;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

lineNo207	POs207_poNo207	Parts207_partNo007	qty207	priceOrdered207	company_207
1	2	P2-1	10	10	X
2	4	P2-2	50	12	Y
NULL	NULL	NULL	NULL	NULL	NULL

- To be able to submit PO and add line, we need to check if ClientID, PartNo match the ones in the system. We also have to check if the quantity the customer is ordering is greater than QOH or not from both company X and Y. If all the conditions are correct, we compare the price of the product from both companies and select the one with the lower price. Afterwards, we query and insert new PO and line:

```
- @app.route('/po', methods=['GET', 'POST'])
```

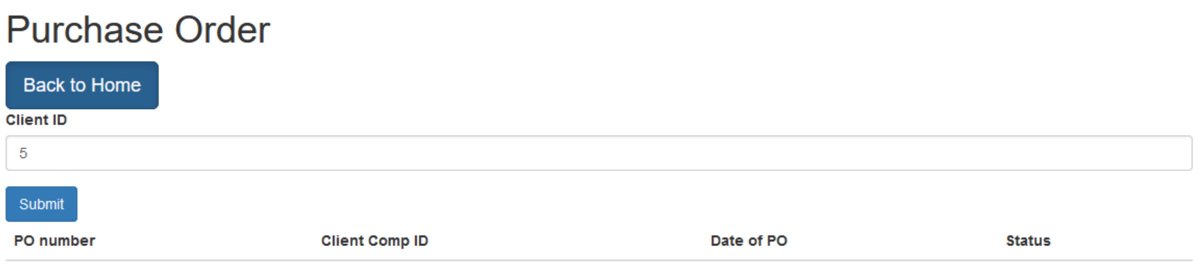
```python
def po():
    #SELECT list of clients
    cur = mysql.connection.cursor()
    cur.execute('''SELECT clientId207 FROM z_db.z_clients207''')
    clientIDList = cur.fetchall()
    print(clientIDList)
    cur.close()

    #SELECT list of part numbers
    cur = mysql.connection.cursor()
    cur.execute('''SELECT partNo007 FROM x_db.x_parts207''')
    partNoList_x = cur.fetchall()
    cur.close()

    cur = mysql.connection.cursor()
    cur.execute('''SELECT partNo007 FROM y_db.y_parts207''')
    partNoList_y = cur.fetchall()
    cur.close()

    if request.method == 'POST':
        compID = request.form['compID']
        clientID = request.form['clientID']
        partNo = request.form['partNo']
        qty = request.form['qty']
        status = 'Pending'
        poNo = 0
        date = datetime.date.today()

        #Check if clientID entered matches the ones in the system
        clientIDCheck = False
        for i in clientIDList:
            for j in i:
                if int(clientID) == int(j):
                    clientIDCheck = True
        #Check if the partNo entered matches the ones in database X
        partNoCheck_x = False
        for i in partNoList_x:
            for j in i:
                if partNo == j:
                    partNoCheck_x = True

         #Check if the partNo entered matches the ones in database y
        partNoCheck_y = False
        for i in partNoList_y:
            for j in i:
```

```
-                       if partNo == j:
-                           partNoCheck_y = True
-
-               #Query to find qoh of part in X
-               cur = mysql.connection.cursor()
-               qohQuery = '''SELECT qoh207 FROM x_db.x_parts207 WHERE partNo007 =
   %s'''
-               cur.execute(qohQuery, [partNo])
-               qohList_x = cur.fetchall()
-               cur.close()
-
-               #Query to find qoh of part in Y
-               cur = mysql.connection.cursor()
-               qohQuery = '''SELECT qoh207 FROM x_db.x_parts207 WHERE partNo007 =
   %s'''
-               cur.execute(qohQuery, [partNo])
-               qohList_y = cur.fetchall()
-               cur.close()
-
-               #Check if qoh is less than quantity ordered or no in x database
-               qohCheck_x = False
-               for i in qohList_x:
-                   for j in i:
-                       if int(qty) < int(j):
-                           qohCheck_x = True
-
-               #Check if qoh is less than quantity ordered or no in y database
-               qohCheck_y = False
-               for i in qohList_y:
-                   for j in i:
-                       if int(qty) < int(j):
-                           qohCheck_y = True
-
-               #Function check if input is correct
-               def checkPoValid(client_ID, qoh, partNo):
-                   if client_ID == True:
-                       if qoh == True and partNo == True:
-                           return True
-
-               #Set company value to send a PO to
-               if checkPoValid(clientIDCheck, qohCheck_x, partNoCheck_x):
-                   company = "X"
-               if checkPoValid(clientIDCheck, qohCheck_y, partNoCheck_y):
-                   company = "Y"
-
```

```
-           #if both company X and Y has provides the same part with sufficent
   qoh
-           if checkPoValid(clientIDCheck, qohCheck_x, partNoCheck_x) and
   checkPoValid(clientIDCheck, qohCheck_y, partNoCheck_y):
-               #Query to find price of entered part in x database
-               cur = mysql.connection.cursor()
-               priceQuery = '''SELECT currentPrice207 FROM x_db.x_parts207
   WHERE partNo007 = %s'''
-               cur.execute(priceQuery, [partNo])
-               priceList_x = cur.fetchall()
-               cur.close()
-
-               price_x = 0
-               for i in priceList_x:
-                   for j in i:
-                       price_x = j
-
-               ##Query to find price of entered part in y database
-               cur = mysql.connection.cursor()
-               priceQuery = '''SELECT currentPrice207 FROM y_db.y_parts207
   WHERE partNo007 = %s'''
-               cur.execute(priceQuery, [partNo])
-               priceList_y = cur.fetchall()
-               cur.close()
-
-               price_y = 0
-               for i in priceList_y:
-                   for j in i:
-                       price_y = j
-
-               #Lower price is used
-               price = min(price_x, price_y)
-
-               #Company value to insert in "lines" table
-               if price_x <= price_y:
-                   company = "X"
-               else:
-                   company = "Y"
-
-           if checkPoValid(clientIDCheck, qohCheck_x, partNoCheck_x) or
   checkPoValid(clientIDCheck, qohCheck_y, partNoCheck_y):
-               #Insert PO to Z database
-               poQuery ="""INSERT INTO z_db.z_pos207 (clientCompID207,
   dataOfPO207, status207, Clients207_clientId207)
-                       VALUES ( %s, %s, %s, %s)"""
-               poValues = ( compID, date, status, clientID)
-               cur = mysql.connection.cursor()
-               cur.execute(poQuery, poValues)
-               mysql.connection.commit()
-               cur.close()
-
-               #Find poNo of the line
-               cur = mysql.connection.cursor()
-               findPoNo = '''SELECT poNo207 FROM z_db.z_pos207 WHERE
   clientCompID207 = %s AND Clients207_clientId207 = %s'''
-               findPoNoValues = (compID, clientID)
-               cur.execute(findPoNo, findPoNoValues)
-               poNoList = cur.fetchall()
-               poNo = poNoList[0]
-
-               #Insert line
-               lineQuery ="""INSERT INTO z_db.z_lines207 (POs207_poNo207,
   Parts207_partNo007, qty207, priceOrdered207, company_207)
-                       VALUES ( %s, %s, %s, %s, %s)"""
-               lineValues = (poNo, partNo, qty, price, company)
-               cur = mysql.connection.cursor()
-               cur.execute(lineQuery, lineValues)
-               mysql.connection.commit()
-               cur.close()
-               return render_template('successPoSubmit.html')
-           else:
-               return render_template('inputError.html')
-
-       return render_template('po.html')
```

- For the last two pages line and poList, they have similar functionality. We want to SELECT data from a table based on a certain condition. On PoList page, we display all the POs that the clientID has:

  (Before)



Purchase Order

Back to Home

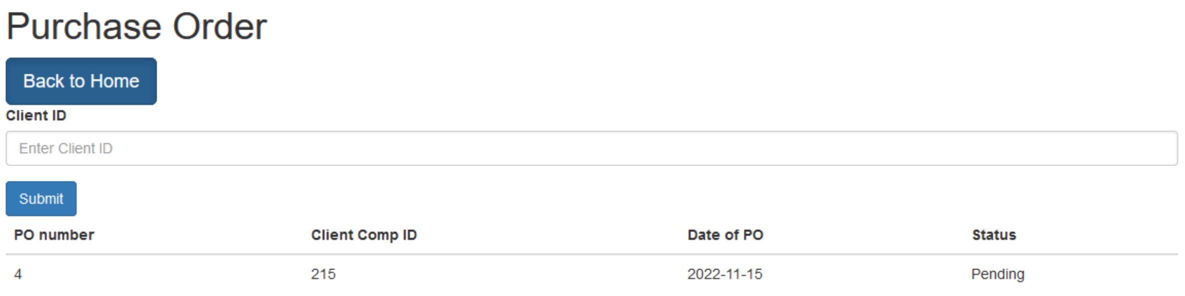
Client ID

5

Submit

PO number	Client Comp ID	Date of PO	Status

(After)



Purchase Order

Back to Home

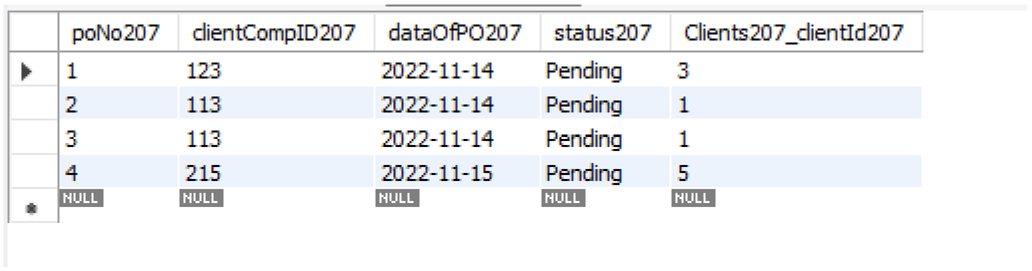
Client ID

Enter Client ID

Submit

PO number	Client Comp ID	Date of PO	Status
4	215	2022-11-15	Pending

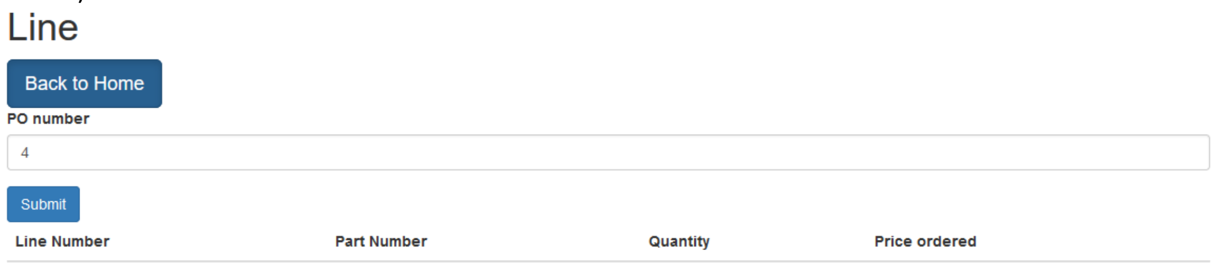
(Table in the database)



poNo207	clientCompID207	dataOfPO207	status207	Clients207_clientId207
1	123	2022-11-14	Pending	3
2	113	2022-11-14	Pending	1
3	113	2022-11-14	Pending	1
4	215	2022-11-15	Pending	5
NULL	NULL	NULL	NULL	NULL

- Similarly, the Line pages does the same thing, but this time with PO number as input and line table as output:

(Before)



Line

Back to Home

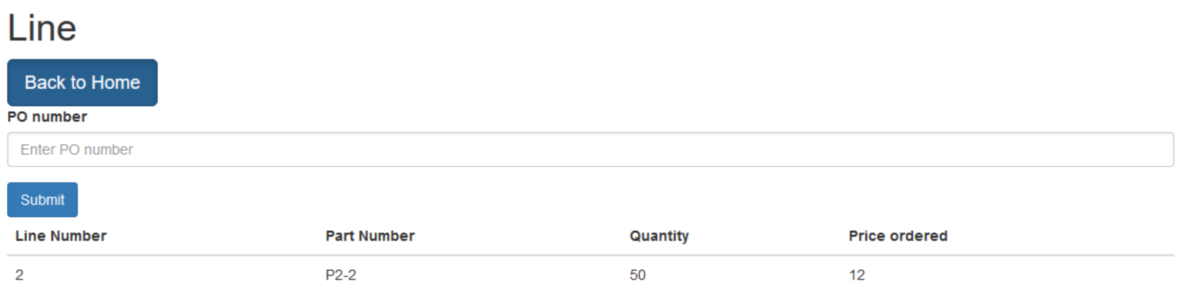
PO number

4

Submit

Line Number	Part Number	Quantity	Price ordered

(After)



Line

Back to Home

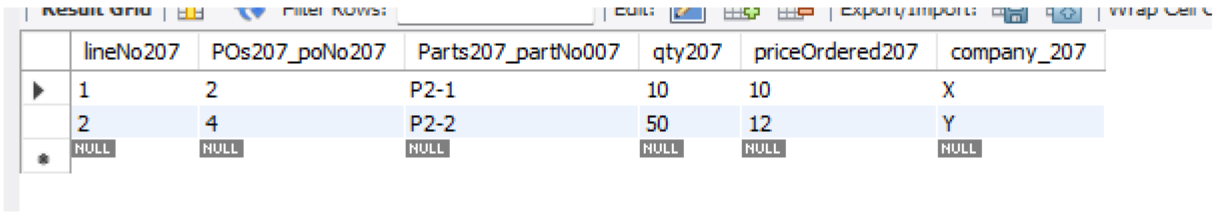
PO number

Enter PO number

Submit

Line Number	Part Number	Quantity	Price ordered
2	P2-2	50	12

(Table in the database)



lineNo207	POs207_poNo207	Parts207_partNo007	qty207	priceOrdered207	company_207
1	2	P2-1	10	10	X
2	4	P2-2	50	12	Y
NULL	NULL	NULL	NULL	NULL	NULL

- Below is the method for poList and Line pages:

```
@app.route('/poList', methods=['GET', 'POST'])
def poList():
    globalData = ''
    if request.method == "POST":
        clientID = request.form.get('clientID')
        cur = mysql.connection.cursor()
        query = """SELECT * FROM z_db.z_pos207 WHERE
Clients207_clientId207 = %s"""
        cur.execute(query, [clientID])
        data = cur.fetchall()
        globalData = data
        cur.close()

    return render_template('poList.html', data=globalData)

@app.route('/line', methods=['GET', 'POST'])
def line():
    globalData = ''
    if request.method == "POST":
        poNo = request.form.get('poNum')
        cur = mysql.connection.cursor()
        query = """SELECT * FROM z_db.z_lines207 WHERE POs207_poNo207 =
%s"""
        cur.execute(query, [poNo])
        data = cur.fetchall()
        globalData = data
        cur.close()

    return render_template('line.html', data=globalData)
```