*# try to accomplish the following in 40 minutes:*

*# create 2 simple tables with different types of variables and auto incrementing primary key*

*# create insert functions for each where all arguments are fed to a python function.*

*# create 2 update functions that change the values inside the tables*

*# create 2 getters that use where logic and conditionals*

*# if you accomplish the above it is 11 points... you need 7.*

*# cheers*

def example\_sql1()

-- Create the database tables

CREATE TABLE IF NOT EXISTS Table1 (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

age INTEGER NOT NULL,

salary REAL

);

CREATE TABLE IF NOT EXISTS Table2 (

id INTEGER PRIMARY KEY AUTOINCREMENT,

category TEXT NOT NULL,

stock INTEGER NOT NULL,

price REAL NOT NULL

);

-- Insert data into Table1

INSERT INTO Table1 (name, age, salary) VALUES ('Alice', 30, 50000.0);

INSERT INTO Table1 (name, age, salary) VALUES ('Bob', 25, 40000.0);

-- Insert data into Table2

INSERT INTO Table2 (category, stock, price) VALUES ('Electronics', 100, 299.99);

INSERT INTO Table2 (category, stock, price) VALUES ('Groceries', 200, 1.99);

-- Update specific fields in Table1

UPDATE Table1

SET age = 31

WHERE id = 1;

-- Update specific fields in Table2

UPDATE Table2

SET stock = 180

WHERE id = 2;

-- Select records from Table1 with conditional logic

SELECT \* FROM Table1

WHERE age >= 26;

-- Select records from Table2 with conditional logic

SELECT \* FROM Table2

WHERE stock >= 150;

1.13 Practice final:   
  
*# try to accomplish the following in 40 minutes:*

*# create 2 simple tables with different types of variables and autoincrementing primary key*

*# create insert functions for each where all arguments are fed to a python function.*

*# create 2 update functions that change the values inside the tables*

*# create 2 getters that use where logic and conditionals*

*# if you accomplish the above it is 11 points... you need 7.*

*# cheers*

from python\_helpers import execute\_sql

def example\_table():

query = '''CREATE TABLE IF NOT EXISTS person (

id INT,

name VARCHAR(25)

);'''

execute\_sql(query, debug=True)

def example\_sql(argA, argB):

query = f"INSERT INTO person (id, name) VALUES ({argA}, '{argB}');"

execute\_sql(query)

def example\_sql2(argA, argB):

query = "INSERT INTO person (id, name) VALUES (%s, %s);"

args = (argA, argB)

execute\_sql(query, args)

if \_\_name\_\_ == "\_\_main\_\_":

execute\_sql('drop table if exists person;')

example\_table()

example\_sql(1, 'bob')

example\_sql(2, 'sally')

example\_sql2(3, 'alice')

print(execute\_sql('select \* from person;'))

from python\_helpers import execute\_sql

# Create two tables with auto-incrementing primary keys

def create\_tables():

# Table 1

query1 = '''

CREATE TABLE IF NOT EXISTS person (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

age INT

);'''

# Table 2

query2 = '''

CREATE TABLE IF NOT EXISTS orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

person\_id INT,

order\_date DATE,

amount DECIMAL(10, 2),

FOREIGN KEY (person\_id) REFERENCES person(id)

);'''

execute\_sql(query1, debug=True)

execute\_sql(query2, debug=True)

# Insert function for person table

def insert\_person(name, age):

query = "INSERT INTO person (name, age) VALUES (%s, %s);"

execute\_sql(query, (name, age))

# Insert function for orders table

def insert\_order(person\_id, order\_date, amount):

query = "INSERT INTO orders (person\_id, order\_date, amount) VALUES (%s, %s, %s);"

execute\_sql(query, (person\_id, order\_date, amount))

# Update function for person table

def update\_person\_age(person\_id, new\_age):

query = "UPDATE person SET age = %s WHERE id = %s;"

execute\_sql(query, (new\_age, person\_id))

# Update function for orders table

def update\_order\_amount(order\_id, new\_amount):

query = "UPDATE orders SET amount = %s WHERE order\_id = %s;"

execute\_sql(query, (new\_amount, order\_id))

# Getter function to fetch person by age

def get\_person\_by\_age(min\_age):

query = "SELECT \* FROM person WHERE age >= %s;"

return execute\_sql(query, (min\_age,))

# Getter function to fetch orders above a certain amount

def get\_orders\_above\_amount(min\_amount):

query = "SELECT \* FROM orders WHERE amount > %s;"

return execute\_sql(query, (min\_amount,))

if \_\_name\_\_ == "\_\_main\_\_":

# Drop tables if they exist

execute\_sql('DROP TABLE IF EXISTS orders;', debug=True)

execute\_sql('DROP TABLE IF EXISTS person;', debug=True)

# Create tables

create\_tables()

# Insert data

insert\_person("Alice", 30)

insert\_person("Bob", 25)

insert\_order(1, "2024-12-01", 150.75)

insert\_order(2, "2024-12-02", 200.50)

# Update data

update\_person\_age(1, 35)

update\_order\_amount(1, 175.00)

# Retrieve and print data

print("Persons aged 25 and above:", get\_person\_by\_age(25))

print("Orders above $175.00:", get\_orders\_above\_amount(175.00))

1.8 Practice final V2

### **Create the following tables and functions.**

### **Players Table**

* **PLAYER\_ID**: Integer, primary key.
* **FIRST\_NAME**: VARCHAR(25).
* **LAST\_NAME**: VARCHAR(50).
* **POSITION**: Integer, range from 0 to 255.
* **ATR1**: Integer, range from 0 to 255.
* **ATR2**: Integer, range from 0 to 255.
* **ATR3**: Integer, range from 0 to 255.

### **Teams Table**

* **TEAM\_ID**: Integer, primary key.
* **NAME**: VARCHAR, unique.
* **MASCOT**: VARCHAR(50), unique.
* **COLOR\_R**: Integer, range from 0 to 255.
* **COLOR\_G**: Integer, range from 0 to 255.
* **COLOR\_B**: Integer, range from 0 to 255.
* **HOMEFIELD**: VARCHAR(100).

### **Names Table**

* **FIRST\_NAME**: VARCHAR(25).
* **LAST\_NAME**: VARCHAR(50).

### **Functions**

1. **Create Player**: A function to create a player and insert the data into the Players table. This function should assign the player a random first\_name, last\_name, ATR1, ATR2, ATR3. POSITION should be set to 255 if not set.
2. **Create Team**: A function to create a team and insert the data into the Team table.
3. **Modify Team:** Modify the Players table to include a foreign key from Team (using Team\_ID). Name this field fk\_team.
4. **Create Player with Team:** same as create player function, but with team included.
5. **Select All Players on a Team**: A function that retrieves all players associated with a specific team.
6. **Transfer Player**: A function to transfer a player from one team to another based on the player's unique ID.

import mysql.connector

import sys

import time

def get\_connection():

num\_of\_tries = 0

wait\_time = 10

while num\_of\_tries < wait\_time:

try:

mydb = mysql.connector.connect(

host="127.0.0.1",

user="root",

db="zybooksdb"

)

if mydb.is\_connected():

print("Connected")

return mydb

except Exception as e:

print("MySQL is not yet running")

num\_of\_tries += 1

print("Waiting for MySQL to start")

time.sleep(1)

if num\_of\_tries >= wait\_time:

sys.exit("MySQL is not starting")

def execute\_sql(sql, arguments = None):

mydb = get\_connection()

mycursor = mydb.cursor()

if arguments is not None:

mycursor.execute(sql, arguments)

else:

mycursor.execute(sql)

if sql.strip().lower().startswith("select"):

results = mycursor.fetchall()

return results

else:

mydb.commit()

mycursor.close()

mydb.close()

def create\_tables():

*# add code here to create your tables*

*# Players*

*# Team*

*# Names*

*# parameters for these tables are given in the assignment description*

pass

def populate\_names\_table():

*# using the csvs first\_names and last\_names populate the names table.*

pass

def create\_player(first\_name = None, last\_name = None, atr1 = None, atr2 = None, atr3 = None, postion = None):

*# SQL statement to insert a player*

*# this function must populate all valid information for a player. Do not assign them to a team at this point.*

sql\_to\_insert\_player = ""

player\_data = ()

execute\_sql(sql\_to\_insert\_player, player\_data)

def create\_team(name,mascot,r,g,b,homefield):

pass

def create\_default\_team()

*# optional used in my debugger*

pass

def create\_player\_with\_team(team, first\_name = None, last\_name = None, atr1 = None, atr2 = None, atr3 = None, position = None):

pass

def select\_all\_players\_on\_team(team\_id):

pass

def modify\_players\_table\_to\_have\_field\_fk\_team():

*# name of the foreign key fk\_team*

*# this must be done after the table is created.*

pass

def transfer\_player(player\_id, team\_id):

*# set the team for player with player\_id to team\_id*

pass

import mysql.connector

def create\_tables():

# SQL statements to create the tables

sql\_players = """

CREATE TABLE IF NOT EXISTS Players (

PLAYER\_ID INT AUTO\_INCREMENT PRIMARY KEY,

FIRST\_NAME VARCHAR(25),

LAST\_NAME VARCHAR(50),

POSITION TINYINT UNSIGNED DEFAULT 255,

ATR1 TINYINT UNSIGNED,

ATR2 TINYINT UNSIGNED,

ATR3 TINYINT UNSIGNED

);

"""

sql\_teams = """

CREATE TABLE IF NOT EXISTS Teams (

TEAM\_ID INT AUTO\_INCREMENT PRIMARY KEY,

NAME VARCHAR(50) UNIQUE,

MASCOT VARCHAR(50) UNIQUE,

COLOR\_R TINYINT UNSIGNED,

COLOR\_G TINYINT UNSIGNED,

COLOR\_B TINYINT UNSIGNED,

HOMEFIELD VARCHAR(100)

);

"""

sql\_names = """

CREATE TABLE IF NOT EXISTS Names (

FIRST\_NAME VARCHAR(25),

LAST\_NAME VARCHAR(50)

);

"""

# Execute the table creation

execute\_sql(sql\_players)

execute\_sql(sql\_teams)

execute\_sql(sql\_names)

def populate\_names\_table(first\_names\_file, last\_names\_file):

import csv

with open(first\_names\_file, 'r') as f\_names, open(last\_names\_file, 'r') as l\_names:

first\_names = [line.strip() for line in csv.reader(f\_names)]

last\_names = [line.strip() for line in csv.reader(l\_names)]

for first\_name in first\_names:

for last\_name in last\_names:

execute\_sql("INSERT INTO Names (FIRST\_NAME, LAST\_NAME) VALUES (%s, %s)", (first\_name[0], last\_name[0]))

def create\_player(first\_name=None, last\_name=None, atr1=None, atr2=None, atr3=None, position=255):

import random

if not first\_name or not last\_name:

result = execute\_sql("SELECT FIRST\_NAME, LAST\_NAME FROM Names ORDER BY RAND() LIMIT 1")

first\_name, last\_name = result[0]

if atr1 is None:

atr1 = random.randint(0, 255)

if atr2 is None:

atr2 = random.randint(0, 255)

if atr3 is None:

atr3 = random.randint(0, 255)

sql = """

INSERT INTO Players (FIRST\_NAME, LAST\_NAME, POSITION, ATR1, ATR2, ATR3)

VALUES (%s, %s, %s, %s, %s, %s);

"""

execute\_sql(sql, (first\_name, last\_name, position, atr1, atr2, atr3))

def create\_team(name, mascot, color\_r, color\_g, color\_b, homefield):

sql = """

INSERT INTO Teams (NAME, MASCOT, COLOR\_R, COLOR\_G, COLOR\_B, HOMEFIELD)

VALUES (%s, %s, %s, %s, %s, %s);

"""

execute\_sql(sql, (name, mascot, color\_r, color\_g, color\_b, homefield))

def modify\_players\_table\_to\_have\_field\_fk\_team():

sql = """

ALTER TABLE Players

ADD COLUMN fk\_team INT,

ADD CONSTRAINT fk\_team FOREIGN KEY (fk\_team) REFERENCES Teams(TEAM\_ID);

"""

execute\_sql(sql)

def create\_player\_with\_team(team\_id, first\_name=None, last\_name=None, atr1=None, atr2=None, atr3=None, position=255):

create\_player(first\_name, last\_name, atr1, atr2, atr3, position)

last\_id = execute\_sql("SELECT LAST\_INSERT\_ID();")[0][0]

execute\_sql("UPDATE Players SET fk\_team = %s WHERE PLAYER\_ID = %s;", (team\_id, last\_id))

def select\_all\_players\_on\_team(team\_id):

sql = """

SELECT \* FROM Players WHERE fk\_team = %s;

"""

return execute\_sql(sql, (team\_id,))

def transfer\_player(player\_id, team\_id):

sql = """

UPDATE Players SET fk\_team = %s WHERE PLAYER\_ID = %s;

"""

execute\_sql(sql, (team\_id, player\_id))

# Ensure the tables are created first

create\_tables()