Assignment 5
DBMS

by
Ayush Agrawal
2023UCS1602

Create Tables

```
import mysql.connector

# Connect to the MySQL database
mydb = mysql.connector.connect(
    host='localhost',
    user='root',
    password='',  # Add your MySQL password if applicable
    database='query run'  # Replace with the actual database name
)

if mydb.is_connected():
    print("Connected to the database")

# Create a cursor object
mycursor = mydb.cursor()

# Create CUSTOMER table
mycursor.execute("""
    CREATE TABLE IF NOT EXISTS CUSTOMER (
        cust_num INT PRIMARY KEY,
        cust_lname VARCHAR(50),
        cust_fname VARCHAR(50),
        cust_balance DECIMAL(10, 2)
    )
""")

# Create PRODUCT table
mycursor.execute("""
    CREATE TABLE IF NOT EXISTS PRODUCT (
        prod_num INT PRIMARY KEY,
        prod_name VARCHAR(100),
        price DECIMAL(10, 2)
    )
""")

# Create INVOICE table
mycursor.execute("""
    CREATE TABLE IF NOT EXISTS INVOICE (
        inv_num INT PRIMARY KEY,
```

```
        prod_num INT,
        cust_num INT,
        inv_date DATE,
        unit_sold INT,
        inv_amount DECIMAL(10, 2),
        FOREIGN KEY (prod_num) REFERENCES PRODUCT(prod_num),
        FOREIGN KEY (cust_num) REFERENCES CUSTOMER(cust_num)
    )
""")

print("Tables created successfully")

# Close the connection
mydb.close()
```

Output

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	customer	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	invoice	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	product	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	3 tables	Sum	0	InnoDB	utf8mb4_general_ci	80.0 KiB	0 B

Insert(Values picked from net)

```
import mysql.connector

# Connect to the MySQL database
mydb = mysql.connector.connect(
    host='localhost',
    user='root',
    password='',  # Add your MySQL password if applicable
    database='query run'  # Replace with your actual database name
)

if mydb.is_connected():
    print("Connected to the database")

# Create a cursor object
mycursor = mydb.cursor()

# Inserting sample data into the CUSTOMER table
customers = [
    (1, 'John', 'Doe', 50000),
    (2, 'Jane', 'Smith', 120000),
    (3, 'Alice', 'Johnson', 0),
    (4, 'Bob', 'Williams', 25000)
]

insert_customer_query = "INSERT INTO CUSTOMER (cust_num, cust_fname, cust_lname,
cust_balance) VALUES (%s, %s, %s, %s)"
mycursor.executemany(insert_customer_query, customers)
mydb.commit()
print("Inserted customers data.")

# Inserting sample data into the PRODUCT table
products = [
    (101, 'Laptop', 75000),
    (102, 'Smartphone', 50000),
    (103, 'Headphones', 5000),
    (104, 'Monitor', 15000)
]
```

```
insert_product_query = "INSERT INTO PRODUCT (prod_num, prod_name, price) VALUES (%s, %s, %s)"
mycursor.executemany(insert_product_query, products)
mydb.commit()
print("Inserted products data.")

# Inserting sample data into the INVOICE table
invoices = [
    (1001, 101, 1, '2023-09-01', 1, 75000),
    (1002, 102, 2, '2023-09-02', 2, 100000),
    (1003, 103, 3, '2023-09-03', 1, 5000),
    (1004, 104, 4, '2023-09-04', 1, 15000)
]

insert_invoice_query = "INSERT INTO INVOICE (inv_num, prod_num, cust_num, inv_date, unit_sold, inv_amount) VALUES (%s, %s, %s, %s, %s, %s)"
mycursor.executemany(insert_invoice_query, invoices)
mydb.commit()
print("Inserted invoices data.")

# Closing the database connection
mydb.close()
```

```
C:\Users\ayush\PycharmProjects\Ayush\.venv\Scripts\python.exe C:\Users\ayush\PycharmProjects\Ayush\insertvalues.py
Connected to the database
Inserted customers data.
Inserted products data.
Inserted invoices data.

Process finished with exit code 0
```

Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM `customer`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	cust_num	cust_lname	cust_fname	cust_balance
Edit Copy Delete	1	Doe	John	50000.00
Edit Copy Delete	2	Smith	Jane	120000.00
Edit Copy Delete	3	Johnson	Alice	0.00
Edit Copy Delete	4	Williams	Bob	25000.00

Server: 127.0.0.1 » Database: query run » Table: invoice

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operati

Showing rows 0 - 3 (4 total, Query took 0.0002 seconds.)

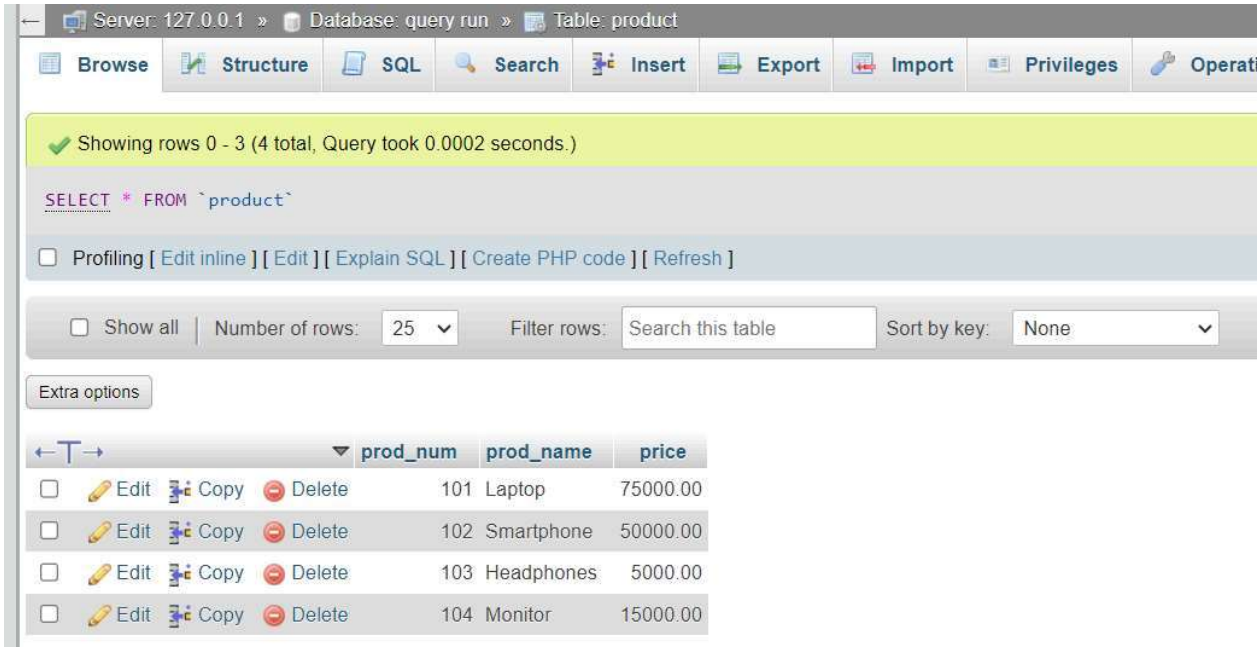
```
SELECT * FROM `invoice`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	inv_num	prod_num	cust_num	inv_date	unit_sold	inv_amount
Edit Copy Delete	1001	101	1	2023-09-01	1	75000.00
Edit Copy Delete	1002	102	2	2023-09-02	2	100000.00
Edit Copy Delete	1003	103	3	2023-09-03	1	5000.00
Edit Copy Delete	1004	104	4	2023-09-04	1	15000.00



Query in order

```
import mysql.connector

# Connect to the MySQL database
mydb = mysql.connector.connect(
    host='localhost',
    user='root',
    password='',  # Add your MySQL password if applicable
    database='query run'  # Replace with the actual database name
)

if mydb.is_connected():
    print("Connected to the database")

# Create a cursor object
mycursor = mydb.cursor()

# a) Find the names of customers who have purchased no item and set cust_balance to 0
print("Query 1: Find customers who haven't purchased any items and set cust_balance = 0")
query1 = """
    SELECT cust_fname, cust_lname
    FROM CUSTOMER
    WHERE cust_num NOT IN (SELECT cust_num FROM INVOICE);
"""
```

```
mycursor.execute(query1)
result1 = mycursor.fetchall()
for row in result1:
    print(f"Customer: {row[0]} {row[1]}")

print("Setting cust_balance = 0 for customers with no purchases...")
query1_update = """
    UPDATE CUSTOMER
    SET cust_balance = 0
    WHERE cust_num NOT IN (SELECT cust_num FROM INVOICE);
"""
mycursor.execute(query1_update)
mydb.commit()

# b) Trigger to update cust_balance when a new invoice record is entered
print("Query 2: Create trigger to update cust_balance when a new invoice record is entered")
query2 = """
    CREATE TRIGGER update_balance
    AFTER INSERT ON INVOICE
    FOR EACH ROW
    BEGIN
        UPDATE CUSTOMER
        SET cust_balance = cust_balance + NEW.inv_amount
        WHERE cust_num = NEW.cust_num;
    END;
"""
mycursor.execute("DROP TRIGGER IF EXISTS update_balance")  # Drop trigger if it exists
mycursor.execute(query2)
print("Trigger created successfully.")

# c) Find customers who purchased more than 3 units of a product in a day
print("Query 3: Find customers who have purchased more than 3 units of a product in a day")
query3 = """
    SELECT cust_fname, cust_lname
    FROM CUSTOMER C
    JOIN INVOICE I ON C.cust_num = I.cust_num
    WHERE I.unit_sold > 3;
"""
mycursor.execute(query3)
result3 = mycursor.fetchall()
```

```
for row in result3:
    print(f"Customer: {row[0]} {row[1]}")

# d) Left, Right, and Full Outer Join examples
print("Query 4: Left Outer Join - Customers with or without invoices")
query4_left = """
    SELECT C.cust_fname, C.cust_lname, I.inv_num
    FROM CUSTOMER C
    LEFT JOIN INVOICE I ON C.cust_num = I.cust_num;
"""
mycursor.execute(query4_left)
result4_left = mycursor.fetchall()
for row in result4_left:
    print(row)

print("\nQuery 5: Right Outer Join - Invoices with or without customers")
query5_right = """
    SELECT C.cust_fname, C.cust_lname, I.inv_num
    FROM CUSTOMER C
    RIGHT JOIN INVOICE I ON C.cust_num = I.cust_num;
"""
mycursor.execute(query5_right)
result5_right = mycursor.fetchall()
for row in result5_right:
    print(row)

print("\nQuery 6: Full Outer Join (via UNION)")
query6_full = """
    SELECT C.cust_fname, C.cust_lname, I.inv_num
    FROM CUSTOMER C
    LEFT JOIN INVOICE I ON C.cust_num = I.cust_num
    UNION
    SELECT C.cust_fname, C.cust_lname, I.inv_num
    FROM CUSTOMER C
    RIGHT JOIN INVOICE I ON C.cust_num = I.cust_num;
"""
mycursor.execute(query6_full)
result6_full = mycursor.fetchall()
for row in result6_full:
    print(row)
```

```
# e) Count number of products sold on each date
print("Query 7: Count number of products sold on each date")
query7 = """
   SELECT inv_date, SUM(unit_sold) AS total_products_sold
   FROM INVOICE
   GROUP BY inv_date;
"""
mycursor.execute(query7)
result7 = mycursor.fetchall()
for row in result7:
   print(f"Date: {row[0]}, Total Products Sold: {row[1]}")

# f) Trigger to copy customer_num to GOLD_CUSTOMER when balance exceeds 100,000
print("Query 8: Create GOLD_CUSTOMER table and trigger")
query8_create_table = """
   CREATE TABLE IF NOT EXISTS GOLD_CUSTOMER (
      cust_num INT PRIMARY KEY
   );
"""
mycursor.execute(query8_create_table)
print("GOLD_CUSTOMER table created.")

query8_trigger = """
   CREATE TRIGGER gold_customer_trigger
   AFTER UPDATE ON CUSTOMER
   FOR EACH ROW
   BEGIN
      IF NEW.cust_balance > 100000 THEN
         INSERT INTO GOLD_CUSTOMER (cust_num)
         VALUES (NEW.cust_num);
      END IF;
   END;
"""
mycursor.execute("DROP TRIGGER IF EXISTS gold_customer_trigger")  # Drop trigger if it
exists
mycursor.execute(query8_trigger)
print("Trigger created successfully.")

# g) Add a new attribute CUST_DOB to the CUSTOMER table
```

```
print("Query 9: Add new column 'CUST_DOB' to CUSTOMER table")
query9 = """
   ALTER TABLE CUSTOMER
   ADD COLUMN cust_dob DATE;
"""
mycursor.execute(query9)
print("CUST_DOB column added successfully.")

# Close the connection
mydb.close()
```

Output

```
C:\Users\ayush\PycharmProjects\Ayush\.venv\Scripts\python.exe
C:\Users\ayush\PycharmProjects\Ayush\Querys.py
Connected to the database
Query 1: Find customers who haven't purchased any items and set cust_balance = 0
Setting cust_balance = 0 for customers with no purchases...
Query 2: Create trigger to update cust_balance when a new invoice record is entered
Trigger created successfully.
Query 3: Find customers who have purchased more than 3 units of a product in a day
Query 4: Left Outer Join - Customers with or without invoices
('John', 'Doe', 1001)
('Jane', 'Smith', 1002)
('Alice', 'Johnson', 1003)
('Bob', 'Williams', 1004)

Query 5: Right Outer Join - Invoices with or without customers
('John', 'Doe', 1001)
('Jane', 'Smith', 1002)
('Alice', 'Johnson', 1003)
('Bob', 'Williams', 1004)

Query 6: Full Outer Join (via UNION)
('John', 'Doe', 1001)
('Jane', 'Smith', 1002)
('Alice', 'Johnson', 1003)
('Bob', 'Williams', 1004)
Query 7: Count number of products sold on each date
Date: 2023-09-01, Total Products Sold: 1
Date: 2023-09-02, Total Products Sold: 2
```

Date: 2023-09-03, Total Products Sold: 1
Date: 2023-09-04, Total Products Sold: 1
Query 8: Create GOLD_CUSTOMER table and trigger
GOLD_CUSTOMER table created.
Trigger created successfully.
Query 9: Add new column 'CUST_DOB' to CUSTOMER table
CUST_DOB column added successfully.

Process finished with exit code 0

Query 1
πcust_fname,cust_lname(CUSTOMER)\πcust_fname,cust_lname(σCUSTOMER.cust_num=INVOICE.cust_num(INVOICE))

Query 2
Not applicable

Query 3

πcust_fname,cust_lname(σunit_sold>3(CUSTOMER⋈INVOICE))

Query 4
Left Outer Join:
πcust_fname,cust_lname,inv_num(CUSTOMER LEFT OUTER JOIN INVOICE)

Query 5
Right Outer Join:
πcust_fname,cust_lname,inv_num(CUSTOMER RIGHT OUTER JOIN INVOICE)

Query 6
Full Outer Join:
(πcust_fname,cust_lname,inv_num(CUSTOMER LEFT OUTER JOIN INVOICE))∪(πcust_fname,cust_lname,inv_num(CUSTOMER RIGHT OUTER JOIN INVOICE))

Query 7
γinv_date,SUM(unit_sold)(INVOICE)

Query 8
Not applicable

Query 9
Not applicable