```
                        # No-SQL_database project

import pymongo

# Establishing the connection to MongoDB
client = pymongo.MongoClient('mongodb+srv://ali:12345@cluster0.k4xb5ra.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0')

# Accessing the database and collection
db = client['database']
collection_customers = db['customers']
collection_orders = db['orders']
collection_users = db['users']
collection_students = db['students']

# Documents to be inserted in customers
customers = [
    {"id": "1", "name": "jery", "city": "lahore"},
    {"id": "2", "name": "harry", "city": "washington"},
    {"id": "3", "name": "john", "city": "konya"},
    {"id": "4", "name": "george", "city": "london"}
]

# Documents to be inserted in orders
orders = [
    {"order_id": "1", "product_name": "phone", "user_id": "1"},
    {"order_id": "2", "product_name": "car", "user_id": "2"},
    {"order_id": "3", "product_name": "watch", "user_id": "3"},
    {"order_id": "4", "product_name": "cloth", "user_id": "4"}
]

# Documents to be inserted in users
users = [
    {"id": "1", "name": "Huxley", "city": "Québec City"},
    {"id": "2", "name": "Oaklee", "city": "Toronto"},
    {"id": "3", "name": "Sutton", "city": "Victoria"},
    {"id": "4", "name": "Jack", "city": "Calgary"}
]

# Documents to be inserted in students
students = [
    {"student_id": "1", "name": "Harper", "Grade": "A", "city": "Edmonton"},
    {"student_id": "2", "name": "Madison", "Grade": "B", "city": "Winnipeg"},
    {"student_id": "3", "name": "Willow", "Grade": "A", "city": "Saskatoon"},
    {"student_id": "4", "name": "Ruby", "Grade": "C", "city": "Windsor"}
]

# Inserting the documents into the collections
collection_customers.insert_many(customers)
collection_orders.insert_many(orders)
collection_users.insert_many(users)
collection_students.insert_many(students)

# Function to count documents in each collection
def count_documents(collection):
    return collection.count_documents({})

# Function to show a sample of 2 documents from each collection
def show_sample_documents(collection):
    return list(collection.find().limit(2))

# Count of documents for each collection
collections = {
    'customers': collection_customers,
    'orders': collection_orders,
    'users': collection_users,
    'students': collection_students
}

for name, collection in collections.items():
    print(f"Count of documents in {name}: {count_documents(collection)}")

# Show sample of 2 documents from each collection
for name, collection in collections.items():
    print(f"\nSample documents from {name}:")
    documents = show_sample_documents(collection)
    for doc in documents:
        print(doc)

# Find a single document
document = collection_students.find_one({"name": "Harper"})
print("\nSingle document found:", document)
```

```
# Find multiple documents
documents = collection_students.find({"name": "Jack"})
print("\nMultiple documents found:")
for doc in documents:
    print(doc)

# Update a single document
collection_students.update_one({"name": "Ruby"}, {"$set": {"Grade": "B"}})

# Update multiple documents
collection_students.update_many({"Grade": "A"}, {"$set": {"Grade": "A+"}})

# Delete a single document
collection_students.delete_one({"name": "Willow"})

# Delete multiple documents
collection_students.delete_many({"Grade": "C"})

# Join operation using local key and foreign key relationship
def show_joined_data(limit):
    # Define the aggregation pipeline for a join
    pipeline = [
        {
            "$lookup": {
                "from": "users",           # The collection to join with
                "localField": "user_id",   # Field from the orders collection
                "foreignField": "id",      # Field from the users collection
                "as": "user_info"          # Name for the resulting array
            }
        },
        {
            "$unwind": {
                "path": "$user_info",
                "preserveNullAndEmptyArrays": True  # Include documents with no matches
            }
        },
        {
            "$lookup": {
                "from": "customers",       # The collection to join with
                "localField": "user_info.name",  # Field from the users collection
                "foreignField": "name",    # Field from the customers collection
                "as": "customer_info"      # Name for the resulting array
            }
        },
        {
            "$unwind": {
                "path": "$customer_info",
                "preserveNullAndEmptyArrays": True  # Include documents with no matches
            }
        },
        {
            "$project": {                  # Select the fields to include in the result
                "user_id": 1,
                "product_name": 1,
                "user_info.name": 1,
                "customer_info.city": 1
            }
        },
        {
            "$limit": limit  # Limit the number of results
        }
    ]

    # Execute the aggregation pipeline
    result = db.orders.aggregate(pipeline)

    # Print the results
    print("\nJoined data from orders, users, and customers:")
    for doc in result:
        print(doc)

# Show the joined data with a limit of 2 results
show_joined_data(limit=2)

# Close the connection
client.close()


              # Results
```

```
Count of documents in customers: 96
Count of documents in orders: 96
Count of documents in users: 96
Count of documents in students: 34

Sample documents from customers:
{'_id': ObjectId('66588f7595f3f03fb8309c67'), 'name': 'lahore', 'city': 'pakistan', 'type': 1}
{'_id': ObjectId('66588f7595f3f03fb8309c68'), 'name': 'ali', 'city': 'pak'}

Sample documents from orders:
{'_id': ObjectId('66588f7795f3f03fb8309c6b'), 'name': 'lahore', 'city': 'pakistan'}
{'_id': ObjectId('66588f7795f3f03fb8309c6c'), 'name': 'ali', 'city': 'pak'}

Sample documents from users:
{'_id': ObjectId('66588f7895f3f03fb8309c6f'), 'name': 'lahore', 'city': 'pakistan'}
{'_id': ObjectId('66588f7895f3f03fb8309c70'), 'name': 'ali', 'city': 'pak'}

Sample documents from students:
{'_id': ObjectId('66596ac21f7c10227b191ee2'), 'student_id': '1', 'name': 'Harper', 'Grade': 'A+', 'city': 'Edmonton'}
{'_id': ObjectId('66596ac21f7c10227b191ee3'), 'student_id': '2', 'name': 'Madison', 'Grade': 'B', 'city': 'Winnipeg'}

Single document found: {'_id': ObjectId('66596ac21f7c10227b191ee2'), 'student_id': '1', 'name': 'Harper', 'Grade': 'A+', 'city': 'Edmont

Multiple documents found:

Joined data from orders, users, and customers:
{'_id': ObjectId('66588f7795f3f03fb8309c6b'), 'user_info': {'name': 'lahore'}, 'customer_info': {'city': 'pakistan'}}
{'_id': ObjectId('66588f7795f3f03fb8309c6b'), 'user_info': {'name': 'lahore'}, 'customer_info': {'city': 'pakistan'}}
```

```
                        # SQL_database project

import sqlite3

# Connect to the SQLite database (or create it if it doesn't exist)
conn = sqlite3.connect('database.db')
cursor = conn.cursor()

# Dropping tables if they exist to ensure a fresh start
cursor.execute('DROP TABLE IF EXISTS users')
cursor.execute('DROP TABLE IF EXISTS orders')
cursor.execute('DROP TABLE IF EXISTS Customers')
cursor.execute('DROP TABLE IF EXISTS Students')

# Creating users tables with schema
cursor.execute('''CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY,
                    name TEXT,
                    age INTEGER)''')

# Creating orders tables with schema
cursor.execute('''CREATE TABLE IF NOT EXISTS orders (
                    order_id INTEGER PRIMARY KEY,
                    user_id INTEGER,
                    product TEXT,
                    FOREIGN KEY(user_id) REFERENCES users(id))''')

# Creating Customers tables with schema
cursor.execute('''CREATE TABLE IF NOT EXISTS Customers (
                    Customer_id INTEGER PRIMARY KEY,
                    Customer_name TEXT,
                    products TEXT,
                    city TEXT,
                    Address TEXT)''')

# Creating Students tables with schema
cursor.execute('''CREATE TABLE IF NOT EXISTS Students (
                    Student_id INTEGER PRIMARY KEY,
                    Student_name TEXT,
                    Student_Roll_no INTEGER,
                    Grade TEXT,
                    Address TEXT)''')

# Inserting sample data into users, orders, Customers and Students
cursor.execute('''INSERT INTO users (name, age) VALUES
                  ('Alice', 30),
                  ('Bob', 24),
                  ('Charlie', 29)''')

# Inserting sample data into orders
cursor.execute('''INSERT INTO orders (user_id, product) VALUES
                  (1, 'Laptop'),
                  (2, 'Smartphone'),
                  (3, 'Tablet')''')
```

```
# Inserting sample data into Customers
cursor.execute('''INSERT INTO Customers (Customer_name, products, city, Address) VALUES
                  ('Alice', 'PC', 'Lahore', 'pk'),
                  ('Bob', 'Mobile_phone', 'Quetta', 'pk'),
                  ('Charlie', 'Tab', 'Sahiwal', 'pk')''')

# Data insertion into Students table
cursor.execute('''INSERT INTO Students (Student_id, Student_name, Student_Roll_no, Grade, Address) VALUES
                  (1, 'John Doe', 101, 'A', '123 Elm St'),
                  (2, 'Jane Doe', 102, 'B', '456 Oak St'),
                  (3, 'Jim Beam', 103, 'C', '789 Pine St')''')

conn.commit()

# Function to count rows in each table
def count_rows(table_name):
    cursor.execute(f'SELECT COUNT(*) FROM {table_name}')
    count = cursor.fetchone()[0]
    return count

# Function to show a sample of 3 rows from each table
def show_sample_rows(table_name):
    cursor.execute(f'SELECT * FROM {table_name} LIMIT 2')
    rows = cursor.fetchall()
    return rows

# Count of rows for each table
tables = ['users', 'orders', 'Customers', 'Students']
for table in tables:
    print(f"Count of rows in {table}: {count_rows(table)}")

# Show sample of 3 rows from each table
for table in tables:
    print(f"\nSample rows from {table}:")
    rows = show_sample_rows(table)
    for row in rows:
        print(row)

# Join operations using primary key and foreign key relationship with LIMIT
def show_inner_join(limit):
    cursor.execute(f'''SELECT users.id, users.name, orders.product, Customers.city
                       FROM users
                       INNER JOIN orders ON users.id = orders.user_id
                       INNER JOIN Customers ON users.name = Customers.Customer_name
                       LIMIT {limit}''')
    rows = cursor.fetchall()
    return rows

def show_left_join(limit):
    cursor.execute(f'''SELECT users.id, users.name, orders.product, Customers.city
                       FROM users
                       LEFT JOIN orders ON users.id = orders.user_id
                       LEFT JOIN Customers ON users.name = Customers.Customer_name
                       LIMIT {limit}''')
    rows = cursor.fetchall()
    return rows

# Simulating RIGHT JOIN using LEFT JOIN by reversing the order of tables
def show_right_join(limit):
    cursor.execute(f'''SELECT users.id, users.name, orders.product, Customers.city
                       FROM orders
                       LEFT JOIN users ON orders.user_id = users.id
                       LEFT JOIN Customers ON users.name = Customers.Customer_name
                       LIMIT {limit}''')
    rows = cursor.fetchall()
    return rows

# Simulating FULL OUTER JOIN using UNION of LEFT JOIN and RIGHT JOIN with LIMIT
def show_full_outer_join(limit):
    cursor.execute(f'''SELECT users.id, users.name, orders.product, Customers.city
                       FROM users
                       LEFT JOIN orders ON users.id = orders.user_id
                       LEFT JOIN Customers ON users.name = Customers.Customer_name
                       UNION
                       SELECT users.id, users.name, orders.product, Customers.city
                       FROM orders
                       LEFT JOIN users ON orders.user_id = users.id
                       LEFT JOIN Customers ON users.name = Customers.Customer_name
                       LIMIT {limit}''')
    rows = cursor.fetchall()
    return rows
```

```
# Show joined data with limit
limit = 3
print("\nInner Join data from users, orders, and Customers:")
inner_joined_rows = show_inner_join(limit)
for row in inner_joined_rows:
    print(row)

print("\nLeft Join data from users, orders, and Customers:")
left_joined_rows = show_left_join(limit=1)
for row in left_joined_rows:
    print(row)

print("\nRight Join data from users, orders, and Customers (simulated):")
right_joined_rows = show_right_join(limit)
for row in right_joined_rows:
    print(row)

print("\nFull Outer Join data from users, orders, and Customers (simulated):")
full_outer_joined_rows = show_full_outer_join(limit=2)
for row in full_outer_joined_rows:
    print(row)

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

```
   # Results

Count of rows in users: 3
Count of rows in orders: 3
Count of rows in Customers: 3
Count of rows in Students: 3

Sample rows from users:
(1, 'Alice', 30)
(2, 'Bob', 24)

Sample rows from orders:
(1, 1, 'Laptop')
(2, 2, 'Smartphone')

Sample rows from Customers:
(1, 'Alice', 'PC', 'Lahore', 'pk')
(2, 'Bob', 'Mobile_phone', 'Quetta', 'pk')

Sample rows from Students:
(1, 'John Doe', 101, 'A', '123 Elm St')
(2, 'Jane Doe', 102, 'B', '456 Oak St')

Inner Join data from users, orders, and Customers:
(1, 'Alice', 'Laptop', 'Lahore')
(2, 'Bob', 'Smartphone', 'Quetta')
(3, 'Charlie', 'Tablet', 'Sahiwal')

Left Join data from users, orders, and Customers:
(1, 'Alice', 'Laptop', 'Lahore')

Right Join data from users, orders, and Customers (simulated):
(1, 'Alice', 'Laptop', 'Lahore')
(2, 'Bob', 'Smartphone', 'Quetta')
(3, 'Charlie', 'Tablet', 'Sahiwal')

Full Outer Join data from users, orders, and Customers (simulated):
(1, 'Alice', 'Laptop', 'Lahore')
(2, 'Bob', 'Smartphone', 'Quetta')
```