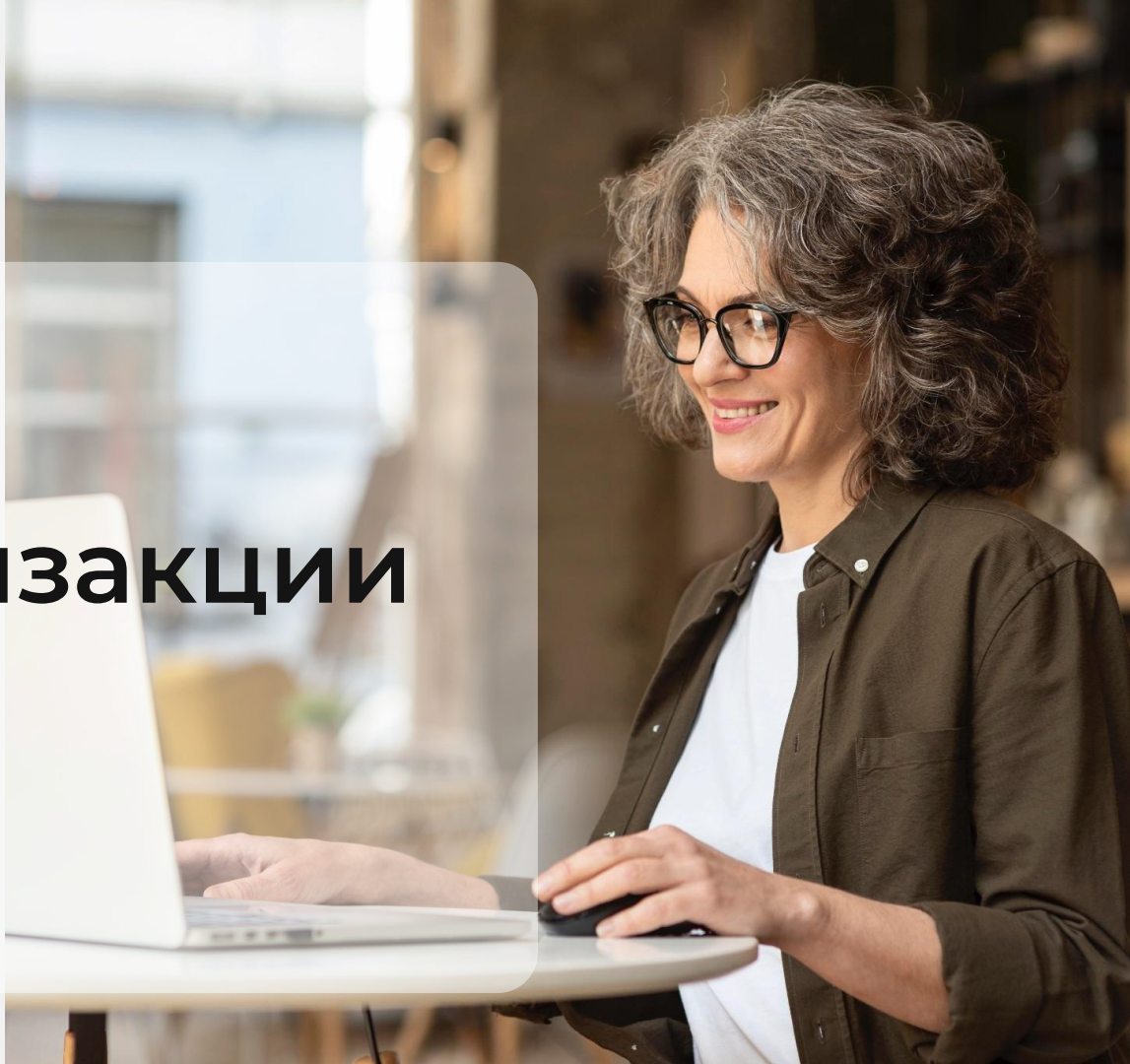


Python

MySQL. Транзакции



Преподаватель

Портрет

Имя Фамилия

Текущая должность

Количество лет опыта

Какой у Вас опыт - ключевые кейсы






Самые яркие проекты

Дополнительная информация по вашему усмотрению









Корпоративный e-mail

Социальные сети (по желанию)

Важно

-  Камера должна быть включена на протяжении всего занятия
-  В течение занятия вопросы задавать в чате или когда преподаватель спрашивает, есть ли у Вас вопросы
-  Вести себя уважительно и этично по отношению к остальным участникам занятия
-  Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях
-  Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя

Повторение

-  Работа с MySQL из Python
-  Подключение к базе данных
-  Работа с курсором
-  Получение результатов запроса
-  Параметризованные запросы
-  Именованные параметры
-  Обработка ошибок
-  Контекстный менеджер

План занятия

- Подключение без указания базы
- Выбор базы данных после подключения
- DictCursor
- Метод commit
- Подключение к серверу с правами на изменения
- Метод executemany
- Транзакции
- Метод rollback



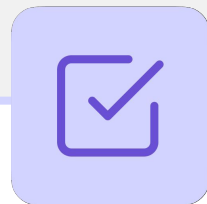
ОСНОВНОЙ БЛОК








Подключение без указания базы

Важно



При подключении к MySQL через `mysql.connect()` **указание базы данных (database=...) не является обязательным.**

Подключение без указания базы данных позволяет:

-  выбрать базу вручную после подключения
-  работать с административными задачами (создание/удаление баз)
-  переключаться между разными базами в рамках одного соединения

Пример

```
import pymysql

config = {'host': 'ich-db.edu.itcareerhub.de',
          'user': 'ich1',
          'password': 'password',
          }

connection = pymysql.connect(**config)

with connection.cursor() as cursor:
    cursor.execute("SHOW DATABASES")
    for db in cursor:
        print(db)
```

Особенности



Команда USE выбирает базу данных на время текущего соединения.



Если база не будет выбрана, запросы к таблицам вызовут ошибку `No database selected`.

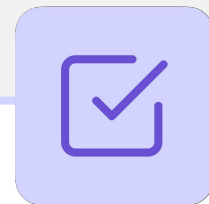


Можно выбирать другую базу в любой момент, выполнив новый USE `<database>`.



DictCursor

Важно



По умолчанию курсор в `mysql` возвращает строки в виде **кортежей**. Чтобы получать результаты в виде словарей `dict` с названиями колонок в качестве ключей, можно использовать `DictCursor`.

Подключение с DictCursor

```
import pymysql
from pymysql.cursors import DictCursor # импорт класса

config = {'host': 'ich-db.edu.itcareerhub.de',
          'user': 'ich1',
          'password': 'password',
          'database': 'hr',
          'cursorclass': DictCursor, # ссылка на класс
        }

connection = pymysql.connect(**config)
```

Пример

```
with connection.cursor() as cursor:
    cursor.execute("SELECT * FROM employees")
    result = cursor.fetchone()
    print(result)
    print(result["first_name"]) # доступ по имени столбца
```



ВОПРОСЫ





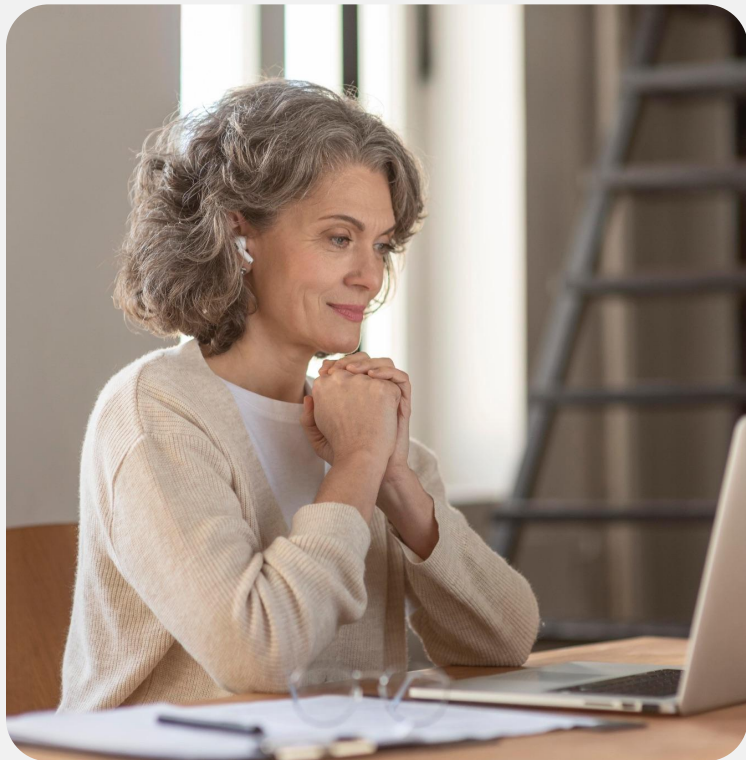
ЗАДАНИЯ





Выберите верные варианты ответа

1. Почему может быть полезно не указывать базу данных при подключении?
 - a. Чтобы сократить время соединения
 - b. Чтобы создавать или удалять базы
 - c. Чтобы ускорить выборку данных
 - d. Чтобы выбирать базу позже вручную



Выберите верные варианты ответа

1. Почему может быть полезно не указывать базу данных при подключении?
 - a. Чтобы сократить время соединения
 - b. Чтобы создавать или удалять базы
 - c. Чтобы ускорить выборку данных
 - d. Чтобы выбирать базу позже вручную



Выберите верный вариант ответа

2. Что произойдёт, если выполнить запрос к таблице без выбора базы данных?

- a. Будет использована первая база
- b. Будет использована стандартная база
- c. Возникнет ошибка No database selected
- d. Вернётся пустой результат



Выберите верный вариант ответа

2. Что произойдёт, если выполнить запрос к таблице без выбора базы данных?

- a. Будет использована первая база
- b. Будет использована стандартная база
- c. Возникнет ошибка No database selected
- d. Вернётся пустой результат



Метод `commit`

Работа с данными в базе



Чтобы **добавить, изменить или удалить данные** в базе, используются запросы INSERT, UPDATE, DELETE.

У них есть важная особенность — данные сначала попадают во **временное хранилище транзакции**.

Запись изменений в базу

Чтобы окончательно записать изменения в базу данных, необходимо вызвать метод

Синтаксис

```
connection.commit()
```

Зачем использовать `commit()`



Без `commit()` изменения **не сохраняются** и будут **потеряны при закрытии соединения**.



Это поведение обеспечивает **безопасность**: изменения применяются только после явного подтверждения.

Создание базы данных и таблиц (`CREATE DATABASE, CREATE TABLE`) не требует вызова `commit()` — эти команды сохраняются сразу.



**Подключение к
серверу с правами на
изменения**

Создание баз и таблиц

Для создания баз и таблиц, необходимо подключиться к серверу с правами на изменения

```
config = {'host': 'ich-edit.edu.itcareerhub.de',  
         'user': 'ich1',  
         'password': 'ich1_password_ilocvedbs',  
         }
```

```
connection = pymysql.connect(**config)
```

Пример: создание базы и таблицы sales

```
with connection.cursor() as cursor:
    # Создание новой базы данных (если ещё не существует)
    cursor.execute("CREATE DATABASE IF NOT EXISTS market")
    cursor.execute("USE market")

    # Создание таблицы продаж
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS sales (
            id INT AUTO_INCREMENT PRIMARY KEY,
            item_name VARCHAR(100),
            quantity INT,
            price DECIMAL(10, 2),
            sale_date DATE
        )
    """)
```

Пример: вставка данных

```
cursor.execute(  
    "INSERT INTO sales (item_name, quantity, price, sale_date) VALUES (%s, %s, %s, %s)",  
    ("Keyboard", 2, 45.50, "2024-06-15")  
)  
connection.commit() # Сохраняем изменения
```

Пример: обновление данных

```
cursor.execute(  
    "UPDATE sales SET quantity = %s WHERE item_name = %s",  
    (3, "Keyboard")  
)  
connection.commit()
```

Пример: удаление данных

```
cursor.execute(  
    "DELETE FROM sales WHERE item_name = %s",  
    ("Keyboard",)  
)  
connection.commit()
```



Метод executemany



Метод `executemany()`

Этот метод используется для **многократного выполнения** одного и того же SQL-запроса **с разными данными**. Это удобно и эффективно при массовом добавлении, обновлении или удалении строк.

Пример

```

cursor.executemany(
    "INSERT INTO sales (item_name, quantity, price, sale_date) VALUES (%s, %s, %s, %s)",
    [
        ("Notebook", 3, 19.99, "2024-06-15"),
        ("Pen", 10, 1.99, "2024-06-16"),
        ("Bag", 1, 49.90, "2024-06-17")
    ]
)
connection.commit()

```

Особенности



Работает только с **изменяющими запросами** (INSERT, UPDATE, DELETE)



Вторым аргументом передаётся **список кортежей**



Экономит ресурсы, т.к. запрос компилируется один раз, а выполняется много



Транзакции



Транзакция

Это **последовательность запросов**, которая выполняется **как единое целое**. Если хотя бы один из шагов завершился ошибкой — **все изменения отменяются**. Это позволяет сохранить **целостность данных** в базе

Преимущества транзакций



Гарантируют **атомарность**: всё или ничего



Защищают от частичных обновлений при сбоях



Полезны при **нескольких связанных изменениях** (например, перемещение денег между счетами)



Метод rollback

Ошибки при транзакции

Если в процессе транзакции возникает ошибка, можно использовать метод `rollback()`.

Этот метод **отменяет все изменения**, сделанные с начала текущей транзакции.

Он используется в связке с `try-except` и `commit()`, чтобы **сохранить целостность данных**.

Если транзакция не подтверждена вызовом `commit()`, то `rollback()` отменяет все действия, выполненные с **момента начала транзакции**

```
connection.rollback()
```

Пример: транзакция покупки товара клиентом

Создадим необходимые таблицы:

- `clients` — список клиентов и их баланс
- `sales` — информация о товарах
- `purchases` — записи о покупках

Затем **попробуем провести оплату**. Клиент попытается купить товар:

- Если у него **хватит денег**, операция завершится успешно.
- Если денег **недостаточно**, база останется без изменений.

Пример: транзакция покупки товара клиентом.

Создание таблиц

1

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    balance DECIMAL(10, 2) NOT NULL CHECK (balance
    >= 0)
)
""")

cursor.execute("""
CREATE TABLE IF NOT EXISTS products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2),
    stock INT NOT NULL CHECK (stock >= 0)
)
""")
```

2

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS purchases (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    purchase_date DATE,
    FOREIGN KEY (customer_id) REFERENCES
    customers(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
)
""")
```

Пример: транзакция покупки товара клиентом.

Заполнение данными

Добавим двух клиентов и два товара. Один клиент с маленьким балансом, другой — с достаточным.

```

cursor.execute("DELETE FROM purchases")
cursor.execute("DELETE FROM customers")
cursor.execute("DELETE FROM products")

cursor.executemany(
    "INSERT INTO customers (name, balance) VALUES (%s, %s)",
    [("Alice", 20.00), ("Bob", 200.00)]
)

cursor.executemany(
    "INSERT INTO products (name, price, stock) VALUES (%s, %s, %s)",
    [("Headphones", 99.99, 3), ("Mouse", 25.00, 5)]
)

connection.commit()

```

Пример: транзакция покупки товара клиентом.

Попытка покупки с недостаточным балансом

Если у клиента недостаточно денег, никакие изменения (списание денег, уменьшение количества товара, запись в покупки) не произойдут. Всё будет отменено.

1

```
try:
```

```
    # Получаем товар
```

```
    cursor.execute("SELECT id, price, stock FROM products WHERE name = %s", ("Headphones",))
    product_id, price, stock = cursor.fetchone()
```

```
    # Если товар в наличии списываем его
```

```
    if stock < 1:
```

```
        raise ValueError("Out of stock")
```

```
    cursor.execute("UPDATE products SET stock = stock - 1 WHERE id = %s", (product_id,))
```

```
    # Получаем клиента
```

```
    cursor.execute("SELECT id, balance FROM customers WHERE name = %s", ("Alice",))
    customer_id, balance = cursor.fetchone()
```

Пример: транзакция покупки товара клиентом.

Попытка покупки с недостаточным балансом

Если у клиента недостаточно денег, никакие изменения (списание денег, уменьшение количества товара, запись в покупки) не произойдут. Всё будет отменено.

2

```
# Если баланса хватает списываем оплату
if balance < price:
    raise ValueError("Insufficient funds")
cursor.execute("UPDATE customers SET balance = balance - %s WHERE id = %s", (price, customer_id))

# Фиксируем покупку в таблице purchases
cursor.execute("INSERT INTO purchases (customer_id, product_id, purchase_date) VALUES (%s, %s,
CURDATE())",
               (customer_id, product_id))

connection.commit() # При отсутствии ошибок завершаем транзакцию
print("Purchase successful.")
except Exception as e:
    connection.rollback() # Откатываем если что-то пошло не так
    print("Transaction failed:", e)
```

Пример: транзакция покупки товара клиентом.

Покупка с достаточным балансом

Здесь у клиента хватает денег, и покупка проходит успешно

```
try:
```

```
# Получаем товар
```

```
cursor.execute("SELECT id, price, stock FROM products WHERE name = %s", ("Mouse",))
product_id, price, stock = cursor.fetchone()
```

```
# Если товар в наличии списываем его
```

```
if stock < 1:
```

```
    raise ValueError("Out of stock")
```

```
cursor.execute("UPDATE products SET stock = stock - 1 WHERE id = %s", (product_id,))
```

```
# Получаем клиента
```

```
cursor.execute("SELECT id, balance FROM customers WHERE name = %s", ("Bob",))
customer_id, balance = cursor.fetchone()
```

1

Пример: транзакция покупки товара клиентом.

Покупка с достаточным балансом

Здесь у клиента хватает денег, и покупка проходит успешно

2

```
# Если баланса хватает списываем оплату
if balance < price:
    raise ValueError("Insufficient funds")
cursor.execute("UPDATE customers SET balance = balance - %s WHERE id = %s", (price, customer_id))

# Фиксируем покупку в таблице purchases
cursor.execute("INSERT INTO purchases (customer_id, product_id, purchase_date) VALUES (%s, %s,
CURDATE())",
               (customer_id, product_id))

connection.commit() # при отсутствии ошибок завершаем транзакцию
print("Purchase successful.")
except Exception as e:
    connection.rollback() # откатываем если что-то пошло не так
    print("Transaction failed:", e)
```



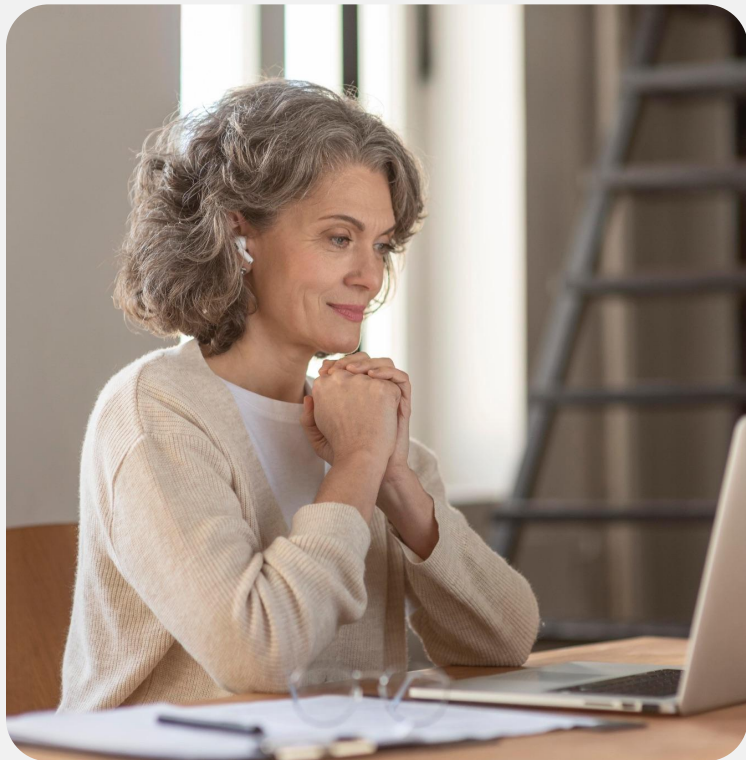
ВОПРОСЫ





ЗАДАНИЯ





Выберите верные варианты ответа

После какого действия нужно вызывать
`commit()`?

- a. SELECT
- b. INSERT
- c. UPDATE
- d. DELETE



Выберите верные варианты ответа

После какого действия нужно вызывать
commit()?

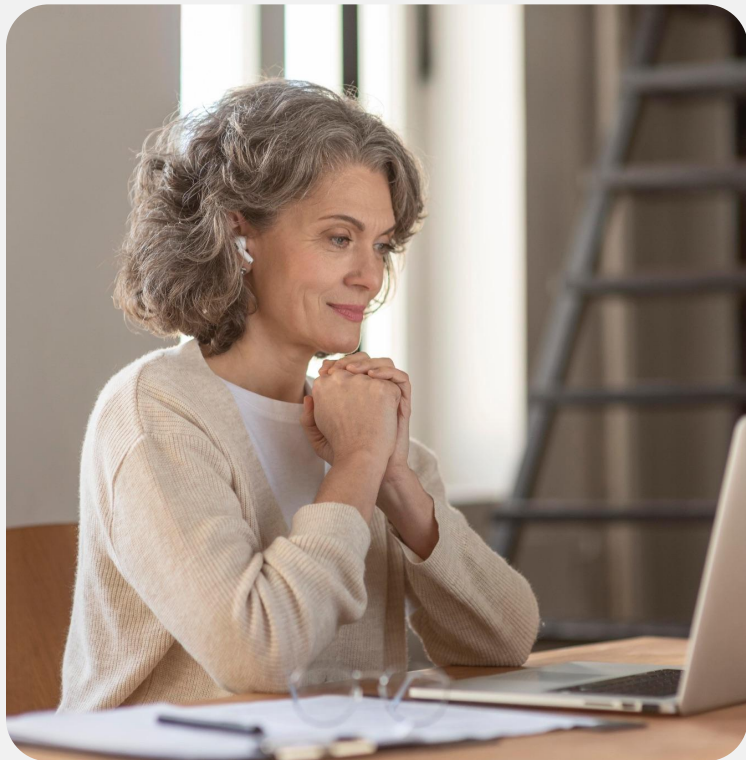
- a. SELECT
- b. INSERT
- c. UPDATE
- d. DELETE



Выберите верный вариант ответа

Что делает метод `executemany()`?

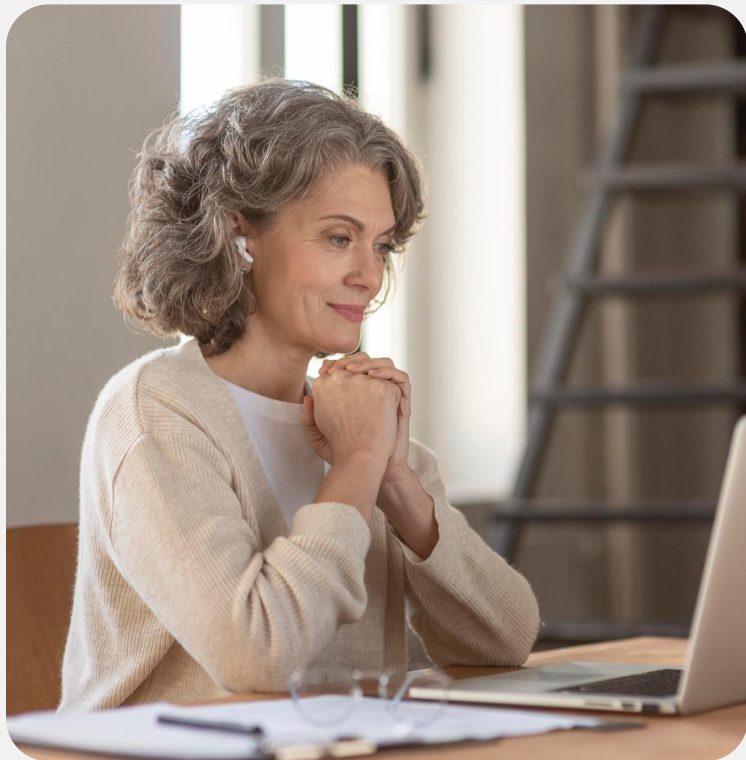
- a. Выполняет SQL-запрос указанное количество раз
- b. Выполняет SQL-запрос указанное количество раз или до первого успешного выполнения
- c. Позволяет выполнять один SQL-запрос с разными данными



Выберите верный вариант ответа

Что делает метод `executemany()`?

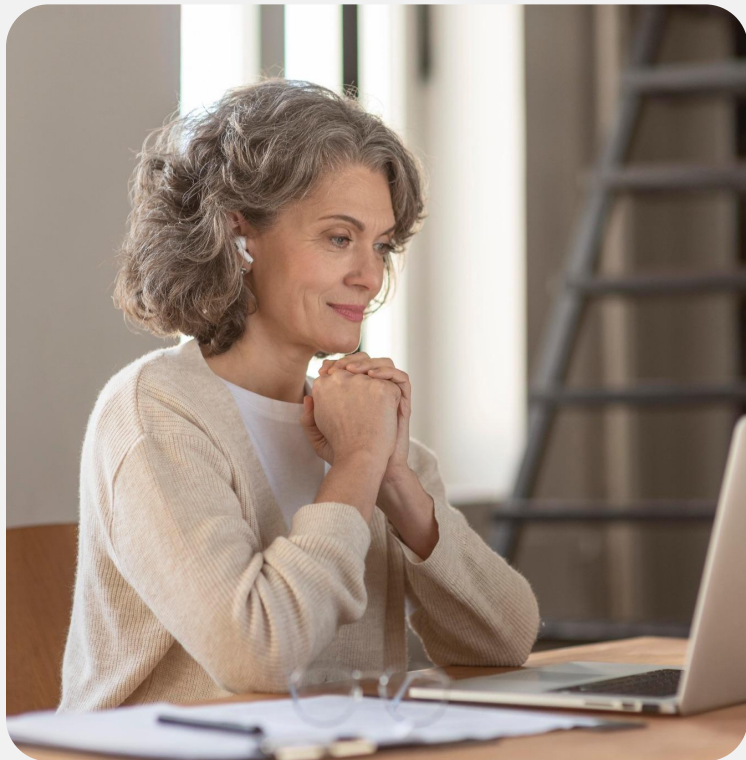
- a. Выполняет SQL-запрос указанное количество раз
- b. Выполняет SQL-запрос указанное количество раз или до первого успешного выполнения
- c. Позволяет выполнять один SQL-запрос с разными данными



Выберите верный вариант ответа

Что произойдёт, если после INSERT не вызвать commit()?

- a. Запрос не выполнится
- b. Изменения будут сохранены автоматически
- c. Изменения будут потеряны после закрытия соединения




Выберите верный вариант ответа

Что произойдёт, если после INSERT не вызвать commit()?

- a. Запрос не выполнится
- b. Изменения будут сохранены автоматически
- c. Изменения будут потеряны после закрытия соединения



ПРАКТИЧЕСКАЯ РАБОТА



Добавление товаров в таблицу

Напишите программу, которая подключается к базе данных market, затем:

- создаёт таблицу products, если она ещё не существует
- добавляет в неё несколько товаров (название и цена)
- выводит список товаров с их ценами

Данные:

```
products = [
    ("Notebook", 10.00),
    ("Pencil", 1.00),
    ("Bag", 25.00)
]
```

Пример вывода:

Products added:

1. Notebook — \$10.00
2. Pencil — \$1.00
3. Bag — \$25.00

Массовое повышение цен

Продолжите предыдущую программу. Теперь:

- увеличьте цену всех товаров на **20%**;
- выполните обновление с помощью одного UPDATE;
- выведите список товаров после изменения цен.

Пример вывода:

Prices updated.

Products after update:

1. Notebook – \$12.00
2. Pencil – \$1.20
3. Bag – \$30.00



ВОПРОСЫ





ДОМАШНЕЕ ЗАДАНИЕ



Домашнее задание

1. Создание базы

Напишите программу, которая:

- создаёт базу данных `notes_app_<your_group>_<your_full_name>`
- выбирает эту базу через `USE notes_app`
- выводит сообщение о результате

Пример вывода:

```
Database 'notes_app' created or already exists.
```

Домашнее задание

2. Добавление заметок

Продолжите предыдущую программу:

- создайте таблицу notes с полями: id, title, content
- вставьте **одну заметку** в таблицу
- выполните `commit()` после вставки
- выведите все заметки используя DictCursor

Пример вывода:

Note added: Shopping `list`

Заключение

