**1. Базовые SQL-запросы (общие для SQLite и SQL Server)**

**1.1 Создание таблицы**

**SQLite**

sql

Copy

Download

CREATE TABLE Users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

age INTEGER,

email TEXT UNIQUE

);

**SQL Server (SSMS)**

sql

Copy

Download

CREATE TABLE Users (

id INT PRIMARY KEY IDENTITY(1,1),

name NVARCHAR(100) NOT NULL,

age INT,

email NVARCHAR(100) UNIQUE

);

**Различия:**

* В SQLite AUTOINCREMENT, в SQL Server IDENTITY(1,1).
* В SQLite TEXT, в SQL Server NVARCHAR.

**1.2 Вставка данных (INSERT)**

**SQLite & SQL Server**

sql

Copy

Download

INSERT INTO Users (name, age, email)

VALUES ('Иван', 25, 'ivan@example.com');

**1.3 Выборка данных (SELECT)**

**SQLite & SQL Server**

sql

Copy

Download

-- Все записи

SELECT \* FROM Users;

-- С условием

SELECT name, age FROM Users WHERE age > 20;

-- Сортировка

SELECT \* FROM Users ORDER BY age DESC;

-- Ограничение вывода (первые 5 записей)

SELECT TOP 5 \* FROM Users; -- SQL Server

SELECT \* FROM Users LIMIT 5; -- SQLite

**1.4 Обновление данных (UPDATE)**

**SQLite & SQL Server**

sql

Copy

Download

UPDATE Users

SET age = 26

WHERE name = 'Иван';

**1.5 Удаление данных (DELETE)**

**SQLite & SQL Server**

sql

Copy

Download

-- Удалить запись

DELETE FROM Users WHERE id = 1;

-- Очистить всю таблицу (но структура останется)

DELETE FROM Users;

-- Полное удаление таблицы

DROP TABLE Users;

**2. Работа с Python (SQLite vs SQL Server)**

**2.1 Подключение к базе данных**

**SQLite (встроен в Python)**

python

Copy

Download

import sqlite3

conn = sqlite3.connect('database.db') # Создаст файл БД

cursor = conn.cursor()

# Создание таблицы

cursor.execute('''

CREATE TABLE IF NOT EXISTS Users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

age INTEGER

)

''')

conn.commit() # Сохраняем изменения

conn.close() # Закрываем соединение

**SQL Server (используем pyodbc)**

python

Copy

Download

import pyodbc

# Подключение к SQL Server

conn = pyodbc.connect(

'DRIVER={SQL Server};'

'SERVER=ВАШ\_СЕРВЕР;'

'DATABASE=ВАША\_БД;'

'UID=ВАШ\_ЛОГИН;'

'PWD=ВАШ\_ПАРОЛЬ'

)

cursor = conn.cursor()

# Создание таблицы

cursor.execute('''

CREATE TABLE Users (

id INT PRIMARY KEY IDENTITY(1,1),

name NVARCHAR(100) NOT NULL,

age INT

)

''')

conn.commit()

conn.close()

**2.2 Вставка и выборка данных через Python**

**SQLite**

python

Copy

Download

import sqlite3

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

# Вставка данных

cursor.execute("INSERT INTO Users (name, age) VALUES (?, ?)", ("Алексей", 30))

conn.commit()

# Выборка данных

cursor.execute("SELECT \* FROM Users")

rows = cursor.fetchall() # Получаем все записи

for row in rows:

print(row)

conn.close()

**SQL Server**

python

Copy

Download

import pyodbc

conn = pyodbc.connect('DRIVER={SQL Server};SERVER=...')

cursor = conn.cursor()

# Вставка данных

cursor.execute("INSERT INTO Users (name, age) VALUES (?, ?)", ("Алексей", 30))

conn.commit()

# Выборка данных

cursor.execute("SELECT \* FROM Users")

rows = cursor.fetchall()

for row in rows:

print(row)

conn.close()

**3. Продвинутые SQL-запросы**

**3.1 JOIN (соединение таблиц)**

**SQLite & SQL Server**

sql

Copy

Download

-- Предположим, есть таблица Orders

CREATE TABLE Orders (

id INT PRIMARY KEY,

user\_id INT,

product TEXT,

FOREIGN KEY (user\_id) REFERENCES Users(id)

);

-- INNER JOIN (только совпадающие записи)

SELECT Users.name, Orders.product

FROM Users

INNER JOIN Orders ON Users.id = Orders.user\_id;

**3.2 Агрегатные функции (GROUP BY)**

**SQLite & SQL Server**

sql

Copy

Download

-- Средний возраст пользователей

SELECT AVG(age) FROM Users;

-- Группировка по возрасту

SELECT age, COUNT(\*) as count

FROM Users

GROUP BY age;

**3.3 Подзапросы (вложенные SELECT)**

**SQLite & SQL Server**

sql

Copy

Download

-- Найти пользователей старше среднего возраста

SELECT name, age

FROM Users

WHERE age > (SELECT AVG(age) FROM Users);

**4. Различия между SQLite и SQL Server**

| **Функция** | **SQLite** | **SQL Server** |
| --- | --- | --- |
| **Типы данных** | TEXT, INTEGER, REAL | NVARCHAR, INT, FLOAT |
| **Автоинкремент** | AUTOINCREMENT | IDENTITY(1,1) |
| **Лимит строк** | LIMIT 10 | TOP 10 |
| **Дата/время** | TEXT (ISO8601) | DATETIME |
| **Подключение** | Файл .db | Сервер (pyodbc) |

**5. Пример полного Python-скрипта для работы с SQLite**

python

Copy

Download

import sqlite3

def create\_db():

conn = sqlite3.connect('shop.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS Products (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

price REAL

)

''')

conn.commit()

conn.close()

def add\_product(name, price):

conn = sqlite3.connect('shop.db')

cursor = conn.cursor()

cursor.execute("INSERT INTO Products (name, price) VALUES (?, ?)", (name, price))

conn.commit()

conn.close()

def get\_products():

conn = sqlite3.connect('shop.db')

cursor = conn.cursor()

cursor.execute("SELECT \* FROM Products")

products = cursor.fetchall()

conn.close()

return products

# Использование

create\_db()

add\_product("Ноутбук", 50000)

add\_product("Телефон", 30000)

products = get\_products()

for product in products:

print(product)

**Вывод**

1. **SQLite** – проще, работает с файлами .db, подходит для небольших проектов.
2. **SQL Server (SSMS)** – мощнее, требует сервер, подходит для корпоративных решений.
3. **Python + SQL** – используйте sqlite3 для SQLite и pyodbc для SQL Server.

Если у вас есть конкретные задания, присылайте — адаптирую код под них! 🚀

Отлично! Ниже — **полная шпаргалка по SQL-запросам**, которые понадобятся тебе на демо-экзамене (для Варианта 3 и Варианта 5). Всё разбито по блокам: **создание таблиц**, **связи**, **вставка данных**, **запросы на выборку**, **расчётные запросы** — с понятными пояснениями и примерами.

# 📚 ШПАРГАЛКА ПО SQL ЗАПРОСАМ

## ✅ 1. СОЗДАНИЕ ТАБЛИЦ

### 🔸 Пример: Таблица "Материалы" (Materials)

CREATE TABLE Materials (

id INTEGER PRIMARY KEY,

name TEXT NOT NULL,

type\_id INTEGER NOT NULL,

price DECIMAL(10,2),

unit TEXT,

quantity\_in\_pack DECIMAL(10,2),

quantity\_in\_stock DECIMAL(10,2),

min\_quantity DECIMAL(10,2),

FOREIGN KEY(type\_id) REFERENCES MaterialTypes(id)

);

📝 **Для чего:**  
Создаёт таблицу материалов. Поле type\_id указывает на тип (внешний ключ).

### 🔸 Пример: Типы материалов (MaterialTypes)

CREATE TABLE MaterialTypes (

id INTEGER PRIMARY KEY,

name TEXT NOT NULL,

loss\_percentage REAL

);

📝 **Для чего:**  
Хранит список типов материала с процентом потерь.

### 🔸 Пример: Продукция (Products)

CREATE TABLE Products (

id INTEGER PRIMARY KEY,

name TEXT NOT NULL,

article TEXT,

type\_id INTEGER,

min\_partner\_price DECIMAL(10,2),

FOREIGN KEY(type\_id) REFERENCES ProductTypes(id)

);

📝 **Для чего:**  
Хранит изделия (продукцию), их тип и минимальную цену для партнёра.

### 🔸 Пример: Заявки партнёров (Applications)

CREATE TABLE Applications (

id INTEGER PRIMARY KEY,

partner\_id INTEGER,

date TEXT,

FOREIGN KEY(partner\_id) REFERENCES Partners(id)

);

📝 **Для чего:**  
Хранит заявки от партнёров, указывает дату и партнёра.

### 🔸 Пример: Продукты в заявке (ApplicationProducts)

CREATE TABLE ApplicationProducts (

application\_id INTEGER,

product\_id INTEGER,

quantity INTEGER,

PRIMARY KEY(application\_id, product\_id),

FOREIGN KEY(application\_id) REFERENCES Applications(id),

FOREIGN KEY(product\_id) REFERENCES Products(id)

);

📝 **Для чего:**  
Хранит, какие продукты и в каком количестве входят в конкретную заявку.

## ➕ 2. ВСТАВКА ДАННЫХ (INSERT)

### 🔸 Вставка одного материала

INSERT INTO Materials (name, type\_id, price, unit, quantity\_in\_pack, quantity\_in\_stock, min\_quantity)

VALUES ('Фанера 12 мм', 1, 570.00, 'лист', 1, 20, 5);

📝 **Для чего:**  
Добавляет новую запись в таблицу Materials.

### 🔸 Вставка типа продукции

INSERT INTO ProductTypes (id, name, coefficient) VALUES (1, 'Шкаф', 1.25);

## 🔍 3. ПРОСТЫЕ SELECT-ЗАПРОСЫ (ПРОСМОТР)

### 🔸 Получить все материалы

SELECT \* FROM Materials;

### 🔸 Получить заявки и партнёров

SELECT a.id, a.date, p.name AS partner\_name

FROM Applications a

JOIN Partners p ON a.partner\_id = p.id;

📝 **Для чего:**  
Показывает список заявок и имена партнёров.

### 🔸 Получить продукты в заявке

SELECT ap.application\_id, pr.name, ap.quantity

FROM ApplicationProducts ap

JOIN Products pr ON ap.product\_id = pr.id

WHERE ap.application\_id = 1;

📝 **Для чего:**  
Выводит продукты, указанные в конкретной заявке (id = 1).

## 💰 4. РАСЧЁТНЫЕ ЗАПРОСЫ

### 🔸 Расчёт стоимости заявки

SELECT SUM(ap.quantity \* p.min\_partner\_price) AS total\_cost

FROM ApplicationProducts ap

JOIN Products p ON ap.product\_id = p.id

WHERE ap.application\_id = 1;

📝 **Для чего:**  
Считает общую сумму заявки, умножая количество на цену.

### 🔸 Расчёт общего количества материала, нужного на все продукты

SELECT mp.material\_id, SUM(mp.required\_quantity \* ap.quantity) AS total\_required

FROM ApplicationProducts ap

JOIN MaterialProducts mp ON ap.product\_id = mp.product\_id

WHERE ap.application\_id = 1

GROUP BY mp.material\_id;

📝 **Для чего:**  
Считает, сколько материалов нужно на всю заявку.

## ⚠️ 5. ПРОВЕРКИ (VALIDATION)

### 🔸 Проверить отрицательные значения

SELECT \* FROM Materials WHERE price < 0 OR min\_quantity < 0;

### 🔸 Найти дубликаты партнёров по email

SELECT email, COUNT(\*) FROM Partners GROUP BY email HAVING COUNT(\*) > 1;

## ✏️ 6. ОБНОВЛЕНИЕ И УДАЛЕНИЕ

### 🔸 Обновление количества на складе

UPDATE Materials SET quantity\_in\_stock = quantity\_in\_stock - 5 WHERE id = 3;

### 🔸 Удаление заявки

DELETE FROM Applications WHERE id = 4;

## 📈 7. СОЗДАНИЕ ВИДА (VIEW)

CREATE VIEW ApplicationView AS

SELECT a.id AS application\_id, p.name AS partner\_name, a.date

FROM Applications a

JOIN Partners p ON p.id = a.partner\_id;

📝 **Для чего:**  
Создаёт виртуальную таблицу для быстрого доступа к заявкам и партнёрам.

## 📐 8. ТЕХНИЧЕСКИЕ ПРИМЕЧАНИЯ

| **Тип данных** | **Что значит** |
| --- | --- |
| INTEGER | Целое число |
| REAL | Вещественное число (дробное) |
| DECIMAL | Точное дробное число (например, цена) |
| TEXT | Строка |
| DATE/TEXT | Дата (для SQLite — лучше как TEXT) |

Если хочешь — могу собрать готовый .sql файл со всеми этими запросами.  
Напиши: **"Сделай SQL-файл"**, и я его сгенерирую.

Готов помочь до конца! 💪