**Список запросов с их описанием.**

**1)Генерация пароля + логин – копирование символов из gmail до @ + новая таблица(содержит id врача, логин, пароль)**

**2) Выбор пациентов, записанных к конкретному врачу**

**3)Выбор всех назначений лекарств, сделанных конкретным врачом**

**4)Выведем полные имена врачей и к какому отделению они относятся, сортирует отделения по алфавиту**

**5)Выведем всех пациентов, которые не посещали врача в течение последних 6 месяцев**

**6) Найти средний возраст пациентов, посещавших каждого врача**

**7) Выведем пациента, список назначенных лекарств, стоимость и общую стоимость**

**8) Все записи на сегодня у определённого врача**

**9) Отдел и число врачей в каждом из них, сортировка по убыванию числа врачей**

**10)Найдём всех врачей, опыт работы которых больше 10 лет**

**11) Найдём пациентов, которые посещали врача в определенный месяц и год**

**12)Найдём врача, у которого не было пациентов за последние 2 недели**

**13)Cумма товара, заказанного у ФармаКо**

**14) Выведем всех больных, которым назначили операцию**

**15) Выведем информацию о всех Запланированных записях**

**16) Счётчик посещений больницы пациентами**

**17) Всех пациентов, которым прописали Витамин Феррумтабс +**

**дозировка**

**18) Выведем пациентов и их рекомендации(лечение), которые были у Офтальмолога**

**19)Выведем тех пациентов, у кого статус записи “Отменено”**

**20)Выведем врачей не зарегистрированных в системе(нет логтина и пароля)**

**Приложение – SQL код всех запросов к вашей базе данных**

**Таблицы:**

CREATE TABLE Patients (

patient\_id SERIAL PRIMARY KEY

full\_name VARCHAR(100) NOT NULL,

gender CHAR(1) CHECK (gender IN ('М', 'Ж')),

date\_of\_birth DATE NOT NULL,

contact\_number VARCHAR(15) UNIQUE NOT NULL,

age INT

);

CREATE TABLE Doctors (

doctor\_id SERIAL PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

contact\_number VARCHAR(15) UNIQUE NOT NULL,

specialization VARCHAR(50) NOT NULL,

gmail VARCHAR(100) UNIQUE NOT NULL,

work\_experience INT

);

CREATE TABLE Appointments (

appointment\_id SERIAL PRIMARY KEY,

patient\_id INT REFERENCES Patients(patient\_id) ON DELETE CASCADE,

doctor\_id INT REFERENCES Doctors(doctor\_id) ON DELETE CASCADE,

appointment\_date TIMESTAMP NOT NULL,

appointment\_status VARCHAR(20) CHECK (appointment\_status IN ('Запланировано', 'Завершено', 'Отменено'))

);

CREATE TABLE Departments (

department\_id SERIAL PRIMARY KEY,

department\_name VARCHAR(100) NOT NULL UNIQUE

);

CREATE TABLE Medicines (

medicine\_id SERIAL PRIMARY KEY,

medicine\_name VARCHAR(100) NOT NULL,

manufacturer VARCHAR(100) NOT NULL,

price DECIMAL(10, 2) CHECK (price >= 0)

);

CREATE TABLE Prescriptions (

prescription\_id SERIAL PRIMARY KEY,

patient\_id INT REFERENCES Patients(patient\_id) ON DELETE CASCADE,

doctor\_id INT REFERENCES Doctors(doctor\_id) ON DELETE CASCADE,

medicine\_id INT REFERENCES Medicines(medicine\_id) ON DELETE CASCADE,

prescription\_date DATE NOT NULL,

dosage VARCHAR(50) NOT NULL

);

CREATE TABLE MedicalRecords (

record\_id SERIAL PRIMARY KEY,

patient\_id INT REFERENCES Patients(patient\_id) ON DELETE CASCADE,

doctor\_id INT REFERENCES Doctors(doctor\_id) ON DELETE CASCADE,

record\_date DATE NOT NULL,

diagnosis TEXT NOT NULL,

treatment TEXT NOT NULL

);

**ЗАПРОСЫ**

1. **Генерация пароля + логин + новая таблица**

CREATE TABLE DoctorLogins (

doctor\_id INT PRIMARY KEY,

login VARCHAR(100) NOT NULL,

password VARCHAR(20) NOT NULL

);

CREATE OR REPLACE FUNCTION generate\_random\_password(length INT DEFAULT 20)

RETURNS TEXT AS $$

DECLARE

chars TEXT := 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&\*()';

password TEXT := '';

i INT;

random\_char CHAR;

BEGIN

FOR i IN 1..length LOOP

random\_char := substr(chars, (random() \* length(chars))::INT + 1, 1);

password := password || random\_char;

END LOOP;

RETURN password;

END;

$$ LANGUAGE plpgsql;

INSERT INTO doctorautor (doctor\_id, login, password)

SELECT

doctor\_id,

SPLIT\_PART(gmail, '@', 1) AS login,

generate\_random\_password(20) AS password

FROM

doctors;

**2)** **Выбор пациентов, записанных к конкретному врачу**

SELECT p.full\_name, a.appointment\_date

FROM patients p

JOIN appointments a ON p.patient\_id = a.patient\_id

WHERE a.doctor\_id = 2 AND a.appointment\_status != 'Завершено';

1. **Выбор всех назначений лекарств, сделанных конкретным врачом**

SELECT p.full\_name, m.medicine\_name, pr.dosage

FROM doctors d

JOIN prescriptions pr ON d.doctor\_id = pr.doctor\_id

JOIN medicines m ON pr.medicine\_id = m.medicine\_id

JOIN patients p ON pr.patient\_id = p.patient\_id

WHERE d.full\_name LIKE 'Гусятин Николай Сергеевич';

1. **Выведем полные имена врачей и к какому отделению они относятся, сортирует отделения по алфавиту**

SELECT d.full\_name, dep.department\_name

FROM doctors d

JOIN departments dep ON d.department\_id = dep.department\_id

ORDER BY dep.department\_name ASC;

1. **Выведем всех пациентов, которые не посещали врача в течение последних 6 месяцев**

SELECT p.patient\_id, p.full\_name

FROM patients p

LEFT JOIN appointments a ON p.patient\_id = a.patient\_id

WHERE a.appointment\_date IS NULL OR a.appointment\_date < NOW() - INTERVAL '6 months';

**6) Найти средний возраст пациентов, посещавших каждого врача**

SELECT d.doctor\_id, d.full\_name, AVG(p.age) AS average\_age

FROM doctors d

JOIN appointments a ON d.doctor\_id = a.doctor\_id

JOIN patients p ON a.patient\_id = p.patient\_id

GROUP BY d.doctor\_id, d.full\_name;

**7) Выведем пациента, список назначенных лекарств, стоимость и общую стоимость**

SELECT p.patient\_id, p.full\_name, m.medicine\_name, m.price, SUM(m.price) AS total\_price

FROM patients p

JOIN prescriptions pr ON p.patient\_id = pr.patient\_id

JOIN medicines m ON pr.medicine\_id = m.medicine\_id

WHERE p.full\_name = 'Петров Андрей Владимирович'

GROUP BY p.patient\_id, m.medicine\_name, m.price;

**8)** **Все записи на сегодня у определённого врача**

SELECT a.appointment\_id, a.appointment\_date, a.appointment\_status, p.patient\_id, p.full\_name AS patient\_name, d.doctor\_id, d.full\_name AS doctor\_name

FROM appointments a

JOIN patients p ON a.patient\_id = p.patient\_id

JOIN doctors d ON a.doctor\_id = d.doctor\_id

WHERE d.full\_name = 'Гусятин Николай Сергеевич' AND DATE(a.appointment\_date) = CURRENT\_DATE AND a.appointment\_status = 'Запланировано';

**9) Отдел и число врачей в каждом из них, сортировка по убыванию числа врачей**

SELECT dp.department\_name, COUNT(d.doctor\_id) AS doctor\_count

FROM departments dp

JOIN doctors d ON d.department\_id = dp.department\_id

GROUP BY dp.department\_name

ORDER BY doctor\_count DESC;

**10)Найдём всех врачей, опыт работы которых больше 10 лет**

SELECT \*

FROM doctors d

WHERE d.work\_experience > 10;

**11) Найдём пациентов, которые посещали врача в определенный месяц и год**

SELECT p.patient\_id, p.full\_name, a.appointment\_date

FROM patients p

JOIN appointments a ON p.patient\_id = a.patient\_id

WHERE EXTRACT(YEAR FROM a.appointment\_date) = 2024 AND EXTRACT(MONTH FROM a.appointment\_date) = 12;

**12)Найдём врача, у которого не было пациентов за последние 2 недели**

SELECT d.doctor\_id, d.full\_name, d.specialization

FROM doctors d

WHERE NOT EXISTS (

SELECT 1

FROM appointments a

WHERE d.doctor\_id = a.doctor\_id

AND a.appointment\_date >= NOW() - INTERVAL '2 weeks'

);

**13)Cумма товара, заказанного у ФармаКо**

SELECT m.manufacturer, (SELECT SUM(price) FROM medicines WHERE manufacturer = 'ФармаКо') AS total\_price

FROM medicines m

WHERE m.manufacturer = 'ФармаКо'

GROUP BY m.manufacturer;

**14) Выведем всех больных, которым назначили операцию**

SELECT p.full\_name, mr.diagnosis ,mr.treatment

FROM medicalrecords mr

JOIN patients p ON p.patient\_id = mr.patient\_id

WHERE mr.treatment LIKE 'Операция';

**15) Выведем информацию о всех Запланированных записях**

SELECT a.appointment\_date, a.appointment\_status, p.full\_name, d.doctor\_id, d.full\_name

FROM appointments a

JOIN patients p ON a.patient\_id = p.patient\_id

JOIN doctors d ON a.doctor\_id = d.doctor\_id

WHERE a.appointment\_status = 'Запланировано';

**16) Счётчик посещений больницы пациентами**

SELECT p.patient\_id, p.full\_name, COUNT(a.appointment\_id) AS visit\_count

FROM patients p

JOIN appointments a ON p.patient\_id = a.patient\_id

GROUP BY p.patient\_id, p.full\_name

ORDER BY visit\_count DESC;

**17)** **Всех пациентов, которым прописали Витамин Феррумтабс + дозировка**

SELECT p.patient\_id, p.full\_name, m.medicine\_name, pr.prescription\_date, pr.dosage

FROM patients p

JOIN prescriptions pr ON p.patient\_id = pr.patient\_id

JOIN medicines m ON pr.medicine\_id = m.medicine\_id

WHERE m.medicine\_name = 'Витамир Феррумтабс';

**18)** **Выведем пациентов и их рекомендации(лечение), которые были у Офтальмолога**

SELECT p.patient\_id, p.full\_name, mr.treatment

FROM patients p

JOIN medicalRecords mr ON p.patient\_id = mr.patient\_id

JOIN doctors d ON mr.doctor\_id = d.doctor\_id

WHERE d.specialization = 'Офтальмолог';

**19)Выведем тех пациентов, у кого статус записи “Отменено”**

SELECT p.patient\_id, p.full\_name, a.appointment\_id, a.appointment\_date, a.appointment\_status

FROM patients p

JOIN appointments a ON p.patient\_id = a.patient\_id

WHERE a.appointment\_status = 'Отменено';

**20)Выведем врачей не зарегистрированных в системе(нет логтина и пароля)**

SELECT d.doctor\_id, d.full\_name, d.specialization

FROM doctors d

LEFT JOIN doctorautor da ON d.doctor\_id = da.doctor\_id

WHERE da.doctor\_id IS NULL;

**Процедуры:**

**1)**CREATE OR REPLACE PROCEDURE add\_appointment(

IN a\_patient\_id INT,

IN a\_doctor\_id INT,

IN a\_appointment\_date TIMESTAMP,

IN a\_appointment\_status VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

INSERT INTO appointments (patient\_id, doctor\_id, appointment\_date, appointment\_status)

VALUES (a\_patient\_id, a\_doctor\_id, a\_appointment\_date, a\_appointment\_status);

END;

$$;

**2)**CREATE OR REPLACE PROCEDURE add\_doctor(

IN d\_full\_name VARCHAR,

IN d\_contact\_number VARCHAR,

IN d\_specialization VARCHAR,

IN d\_gmail VARCHAR,

IN d\_work\_experience INT

)

LANGUAGE plpgsql

AS $$

BEGIN

INSERT INTO doctors (full\_name, contact\_number, specialization, gmail, work\_experience)

VALUES (d\_full\_name, d\_contact\_number, d\_specialization, d\_gmail, d\_work\_experience);

END;

$$;

**3)**CREATE OR REPLACE PROCEDURE add\_patient(

IN p\_full\_name VARCHAR,

IN p\_gender CHAR,

IN p\_date\_of\_birth DATE,

IN p\_contact\_number VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

INSERT INTO patients (full\_name, gender, date\_of\_birth, contact\_number)

VALUES (p\_full\_name, p\_gender, p\_date\_of\_birth, p\_contact\_number);

END;

$$;

**4)** CREATE OR REPLACE PROCEDURE update\_doctor\_departments()

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE doctors d

SET department\_id = (

SELECT department\_id

FROM departments

WHERE department\_name = d.specialization || ‘ия’

);

END;

$$;

**5)** CREATE OR REPLACE PROCEDURE add\_medical\_record(

IN c\_patient\_id INT,

IN c\_doctor\_id INT,

IN c\_record\_date DATE,

IN c\_diagnosis TEXT,

IN c\_treatment TEXT

)

LANGUAGE plpgsql

AS $$

BEGIN

-- Проверка, что встреча завершена и дата соответствует

IF EXISTS (

SELECT 1

FROM appointments

WHERE patient\_id = c\_patient\_id

AND doctor\_id = c\_doctor\_id

AND appointment\_status = 'Завершено'

AND DATE(appointment\_date) = c\_record\_date

) THEN

INSERT INTO medicalrecords (patient\_id, doctor\_id, record\_date, diagnosis, treatment)

VALUES (c\_patient\_id, c\_doctor\_id, c\_record\_date, c\_diagnosis, c\_treatment);

ELSE

RAISE EXCEPTION 'Невозможно добавить запись: встреча не завершена или дата не соответствует';

END IF;

END;

$$;

**6)** CREATE OR REPLACE PROCEDURE add\_prescription(

IN p\_patient\_id INT,

IN p\_doctor\_id INT,

IN p\_medicine\_id INT,

IN p\_prescription\_date DATE,

IN p\_dosage VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

-- Проверка, что медицинская запись существует и статус обследования "Завершено"

IF EXISTS (

SELECT 1

FROM MedicalRecords mr

JOIN Appointments a ON mr.patient\_id = a.patient\_id AND mr.doctor\_id = a.doctor\_id

WHERE mr.patient\_id = p\_patient\_id

AND mr.doctor\_id = p\_doctor\_id

AND mr.record\_date = p\_prescription\_date

AND a.appointment\_status = 'Завершено'

) THEN

INSERT INTO Prescriptions (patient\_id, doctor\_id, medicine\_id, prescription\_date, dosage)

VALUES (p\_patient\_id, p\_doctor\_id, p\_medicine\_id, p\_prescription\_date, p\_dosage);

ELSE

RAISE EXCEPTION 'Невозможно добавить рецепт: медицинская запись не существует или статус обследования не "Завершено"';

END IF;

END;

$$;

**Триггеры:**

**1)** CREATE OR REPLACE FUNCTION check\_appointment\_gender()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.doctor\_id = (SELECT doctor\_id FROM doctors WHERE specialization = 'Уролог') AND NEW.patient\_id = (SELECT patient\_id FROM patients WHERE gender = 'Ж') THEN

RAISE EXCEPTION 'Женщины не могут записываться к урологу';

ELSIF NEW.doctor\_id = (SELECT doctor\_id FROM Doctors WHERE specialization = 'Гинеколог') AND NEW.patient\_id = (SELECT patient\_id FROM patients WHERE gender = 'М') THEN

RAISE EXCEPTION 'Мужчины не могут записываться к гинекологу';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_appointment\_gender\_trigger

BEFORE INSERT OR UPDATE ON appointments

FOR EACH ROW

EXECUTE FUNCTION check\_appointment\_gender();

**2)** CREATE OR REPLACE FUNCTION check\_appointment\_time()

RETURNS TRIGGER AS $$

DECLARE

existing\_appointment RECORD;

BEGIN

SELECT 1

INTO existing\_appointment

FROM appointments

WHERE doctor\_id = NEW.doctor\_id

AND appointment\_date = NEW.appointment\_date

AND appointment\_status != 'Отменено';

IF FOUND THEN

RAISE EXCEPTION 'Запись на это время уже существует';

END IF;

SELECT 1

INTO existing\_appointment

FROM appointments

WHERE doctor\_id = NEW.doctor\_id

AND appointment\_status != 'Отменено'

AND (

(appointment\_date BETWEEN NEW.appointment\_date - INTERVAL '30 minutes' AND NEW.appointment\_date + INTERVAL '30 minutes')

OR (NEW.appointment\_date BETWEEN appointment\_date - INTERVAL '30 minutes' AND appointment\_date + INTERVAL '30 minutes')

);

IF FOUND THEN

RAISE EXCEPTION 'Запись на это время невозможна, так как существует другая запись в пределах 30 минут';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_appointment\_time\_trigger

BEFORE INSERT OR UPDATE ON Appointments

FOR EACH ROW

EXECUTE FUNCTION check\_appointment\_time();

**3)** CREATE OR REPLACE FUNCTION update\_patient\_age()

RETURNS TRIGGER AS $$

BEGIN

NEW.age := EXTRACT(YEAR FROM AGE(NEW.date\_of\_birth));

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_age

BEFORE INSERT OR UPDATE ON patients

FOR EACH ROW

EXECUTE FUNCTION update\_patient\_age();

**Функция:**

CREATE OR REPLACE FUNCTION generate\_random\_password(length INT DEFAULT 20)

RETURNS TEXT AS $$

DECLARE

chars TEXT := 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&\*()';

password TEXT := '';

i INT;

random\_char CHAR;

BEGIN

FOR i IN 1..length LOOP

random\_char := substr(chars, (random() \* length(chars))::INT + 1, 1);

password := password || random\_char;

END LOOP;

RETURN password;

END;

$$ LANGUAGE plpgsql;