Отчет по Проекту Базы Данных – АВТОШКОЛА

Оглавление

[**1-2. Структура БД и ER-модель** 2](#_Toc199286857)

[**3. Список созданных запросов с описанием:** 5](#_Toc199286858)

[**4.** **Список процедур, функций, триггеров** 7](#_Toc199286859)

[**5. Рекомендации по работе с БД** 7](#_Toc199286860)

[**6. Описание работы с GUI** 7](#_Toc199286861)

[**7. Приложение: SQL-код** 8](#_Toc199286862)

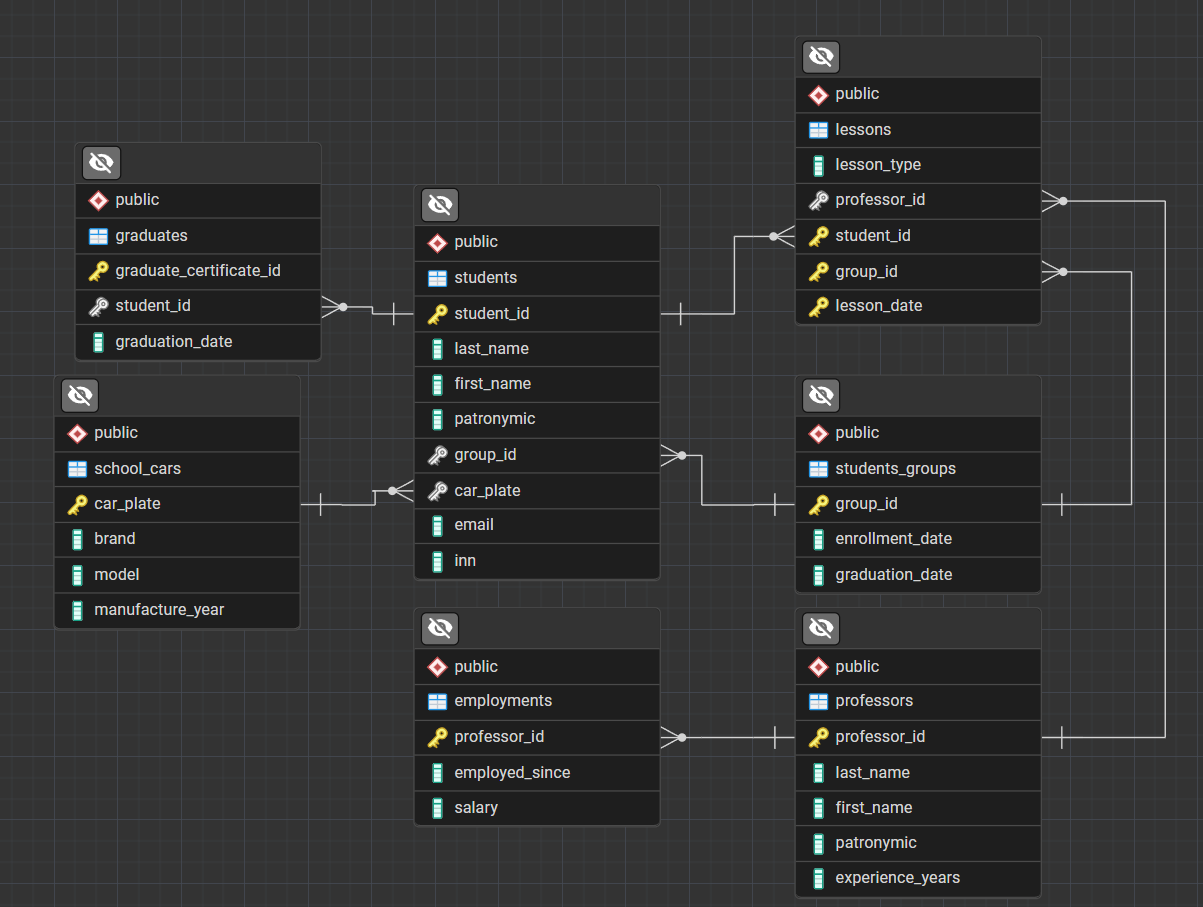
[ GitHub 8](#_Toc199286863)

[ create.sql 8](#_Toc199286864)

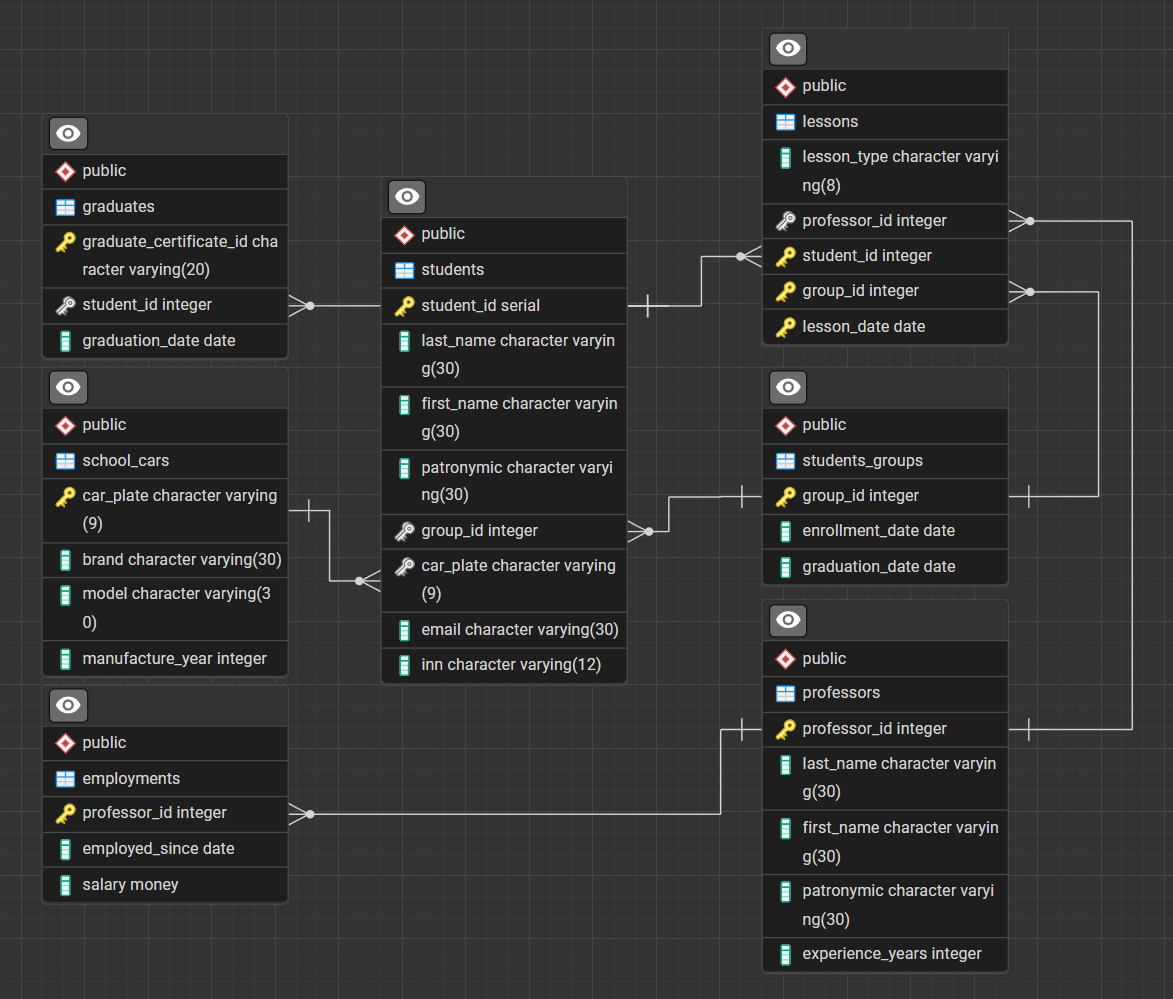
[ insert.sql 20](#_Toc199286865)

[ queries.sql 36](#_Toc199286866)

# **1-2. Структура БД и ER-модель**



ER-модель базы данных «Автошкола» без указания типов данных полей



ER-модель базы данных «Автошкола» с указанием типов данных полей

База данных представлена 7 (семью) таблицами. Среди этих таблиц:

* Ученики
* Преподаватели
* Занятия
* Группы учеников
* Трудоустройства (преподавателей)
* Учебные авто
* Список выпускников (с номером выпускного сертификата)

В структуре реализованы связи:

* Students – Lessons (Один-ко-многим)
* Professors – Lessons (Один-ко-многим)
* Employments – Professors (Один-к-одному)
* Students\_groups – Students (Один-ко-многим)
* Students\_groups – Lessons (Один-ко-многим)
* School\_cars – Students (Один-ко-многим)
* Students – Graduates (Один-к-одному) (допускается 1-к-0)

Пользоваться БД может администратор автошколы, бухгалтер.

- Учащиеся могут, обратившись к БД, уточнить номер своего учебного авто.

- Выпускники – уточнить номер выпускного сертификата об окончании автошколы.

- Администратор может воспользоваться процедурами для добавления новых групп, студентов, выпускников автошколы.

Также, среди ограничений БД:

* В случае, если у студента нету группы – значение поля группа принимается за «-1». Как следствие, в целях целостности отношений в РБД существует триггер, который гарантирует наличие такового ключа.
* Со всеми остальными ограничениями на поля таблиц можно ознакомиться в коде (напр. Номера соответствуют шаблону номеров РФ).

# **3. Список созданных запросов с описанием:**

№1

Выбрать ФИО всех студенток из всех нечетных групп, зачисленных после 31.12.2024

№2

Вывести всех студентов, чья почта не зарегистрированна на @gmail.com или mail.ru.

Отсортировать по дате поступления (сначала новые, потом уже те, что раньше поступили),

ФИО (По Возрастающей)

№3

Вывести всех преподавателей с окладом >100.000. Отсортировать по Окладу по убывающей

№4

Вывести количество преподавателей к каждой "вилке" окладов с шагом 30000 рублей

№5

Найти среднюю ЗП преподавателей НЕ УЧИТЫВАЯ 2 самые высокие и 2 самые низкие

№6

Вывести студентов, у которых ВНУТРИ имени почты содержится латинская букву A,

отсортировать по фамилиям в обратном порядке.

№7

Вывести всех преподавателей, у которых свыше 15 лет стажа

и при этом они трудоустроены до лета 2024 года.

№8

Вывести все групповые занятия (без повторений) с датой зачисления группы.

№9

Вывести всех преподавателей с ЗП выше средней по Автошколе.

Первой строкой также вывести среднюю ЗП.

№10

Вывести, сколько и каких занятий провел каждый преподаватель

№11

Студентов, прошедших как теорию, так и практику

№12

Всех выпускников, что не являются преподавателями в нашей Автошколе

№13

Студентов, которым преподавал один и тот же профессор и теорию, и практику

№14

Топ 3 самых частых учебных авто у студентов

№15

Вывести преподавателей, у которых было больше 5 практических занятий за май 2025 года

№16

Вывести 3 студентов, у которых было большего всего практических занятий

и 3 с наименьшим их числом

№17

Вывести все группы, в которых есть иностранные студенты (нету отчеств)

№18

Вывести дату последнего практического занятия у каждого студента и количество этих занятий

№19

Вывести id студентов, что занимались с одним и тем же преподавателем 2 или более раз

№20

Вывести студентов, у которых занятия вёл хотя бы 1 из 3 преподавателей

с самой большой ЗП

# **Список процедур, функций, триггеров**

* Триггер, для автоматического заполнения столбца со значением идентификатора сертификата об окончании автошколы
* То есть перед каждым добавлением выпускника в таблицу выпускников мы берем эту новую запись, ставим сегодняшнюю дату как дату получения диплома об окончании, а далее генерируем идентификатор сертификата по шаблону
* Триггер для того, чтобы группам студентов и занятиям, у которых при добавлении не была указана группа приписывалась группа «-1»
* Триггер для того, чтобы в таблице students\_groups всегда была группа с id = -1 (для студентов без группы) в целях поддержания целостности данных (чтобы не было ошибок по внешнему ключу
* Триггер, чтобы даже при удалении всех строк и таблицы групп оставалась эта пустая группа с id= «-1» (использует процедуру, так как функция не может применять вставку)
* Процедуры для добавления студента в таблицу выпускников на два случая – с указанием конкретной даты и без (ставится сегодняшняя)
* Процедуры для добавления группы в таблицу групп на два случая – с указанием конкретной даты зачисления и без (ставится сегодняшняя)
* Процедуры для добавления студента в таблицу студентов

# **5. Рекомендации по работе с БД**

В целях ускорения поиска студентов на наличие посещений занятий был создан индекс в таблице lessons на поле student\_id.

# **6. Описание работы с GUI**

To be released soon...

# **7. Приложение: SQL-код**

# GitHub

В целях удобства прикладываю также SQL-код отдельными файлами. А также прикладываю ссылку на GitHub с актуальным кодом: <https://github.com/Thunboo/MIET_IB/tree/main/4th%20Term/DataBases/Project>

# create.sql

-- Создаём таблицу для Учебных авто

CREATE TABLE school\_cars (

car\_plate varchar(9) NOT NULL,

brand varchar(30) NOT NULL,

model varchar(30) NOT NULL,

manufacture\_year int

);

-- Создаём таблицу для сертификатов выпускников Автошколы

CREATE TABLE graduates (

graduate\_certificate\_id varchar(20),

student\_id int,

graduation\_date date NOT NULL

);

-- Создаём таблицу Преподавателей и Инструкторов

CREATE TABLE professors (

professor\_id int NOT NULL,

last\_name varchar(30) NOT NULL,

first\_name varchar(30) NOT NULL,

patronymic varchar(30),

experience\_years int NOT NULL

);

-- Создаём таблицу трудоустроств Преподавателей и Инструкторов

CREATE TABLE employments (

professor\_id int,

employed\_since date,

salary money

);

-- Создаём таблицу под группы учащихся

CREATE TABLE students\_groups (

group\_id int, --NOT NULL,

enrollment\_date date DEFAULT CURRENT\_DATE NOT NULL,

graduation\_date date DEFAULT (CURRENT\_DATE + '3 months'::interval) NOT NULL

);

-- Создаём таблицу Учащихся

CREATE TABLE students (

student\_id SERIAL NOT NULL,

last\_name varchar(30) NOT NULL,

first\_name varchar(30) NOT NULL,

patronymic varchar(30),

group\_id int DEFAULT NULL,

car\_plate varchar(9) NOT NULL DEFAULT 'О000ОО00',

email varchar(30),

inn varchar(12) DEFAULT '770000000000'::character varying

);

-- Создаём таблицу занятий (связь профессор-студент)

CREATE TABLE lessons (

lesson\_type varchar(8),

professor\_id int NOT NULL,

student\_id int NOT NULL,

group\_id int,

lesson\_date date DEFAULT CURRENT\_DATE NOT NULL

);

-- Ограничения -------------------------------

-- school\_cars

ALTER TABLE ONLY school\_cars

ADD CONSTRAINT school\_cars\_pkey PRIMARY KEY (car\_plate);

ALTER TABLE ONLY school\_cars

ADD CONSTRAINT school\_cars\_car\_plate\_check CHECK (car\_plate ~\* '^[АВЕКМНОРСТУХ]{1}[0-9]{3}[АВЕКМНОРСТУХ]{2}[0-9]{2,3}$');

ALTER TABLE ONLY school\_cars

ADD CONSTRAINT school\_cars\_manufacture\_year\_check

CHECK (manufacture\_year > 1990 AND manufacture\_year <= EXTRACT(year from CURRENT\_DATE)::int);

-- students\_groups

ALTER TABLE ONLY students\_groups

ADD CONSTRAINT students\_groups\_pkey PRIMARY KEY (group\_id);

-- students

ALTER TABLE ONLY students

ADD CONSTRAINT students\_pkey PRIMARY KEY (student\_id);

ALTER TABLE ONLY students

ADD CONSTRAINT students\_students\_groups\_fkey FOREIGN KEY (group\_id) REFERENCES students\_groups(group\_id) ON DELETE SET DEFAULT;

ALTER TABLE ONLY students

ADD CONSTRAINT students\_cars\_fkey FOREIGN KEY (car\_plate) REFERENCES school\_cars(car\_plate) ON DELETE SET DEFAULT;

ALTER TABLE ONLY students

ADD CONSTRAINT students\_email\_check CHECK ((email)::text ~\* '^[A-Za-z0-9.\_+%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'::text);

-- graduates

ALTER TABLE ONLY graduates

ADD CONSTRAINT graduates\_pkey PRIMARY KEY (graduate\_certificate\_id);

ALTER TABLE ONLY graduates

ADD CONSTRAINT graduates\_graduation\_date\_check CHECK (graduation\_date < CURRENT\_DATE + '1 days'::interval);

ALTER TABLE ONLY graduates

ADD CONSTRAINT graduates\_student\_id\_fkey FOREIGN KEY (student\_id) REFERENCES students(student\_id) ON DELETE CASCADE;

-- professors

ALTER TABLE ONLY professors

ADD CONSTRAINT professors\_pkey PRIMARY KEY (professor\_id);

ALTER TABLE ONLY professors

ADD CONSTRAINT professors\_experience\_check CHECK (experience\_years > 3);

-- employments

ALTER TABLE ONLY employments

ADD CONSTRAINT employments\_pkey PRIMARY KEY (professor\_id);

ALTER TABLE ONLY employments

ADD CONSTRAINT employments\_professor\_id\_fkey FOREIGN KEY (professor\_id) REFERENCES professors(professor\_id) ON DELETE CASCADE;

ALTER TABLE ONLY employments

ADD CONSTRAINT employments\_employed\_since\_check CHECK (employed\_since < CURRENT\_DATE + '15 days'::interval);

-- lessons

ALTER TABLE ONLY lessons

ADD CONSTRAINT lessons\_pkey PRIMARY KEY (student\_id, group\_id, lesson\_date);

ALTER TABLE ONLY lessons

ADD CONSTRAINT lessons\_professor\_id\_fkey FOREIGN KEY (professor\_id) REFERENCES professors(professor\_id);-- ON DELETE CASCADE;

ALTER TABLE ONLY lessons

ADD CONSTRAINT lessons\_student\_id\_fkey FOREIGN KEY (student\_id) REFERENCES students(student\_id) ON DELETE CASCADE;

ALTER TABLE ONLY lessons

ADD CONSTRAINT lessons\_group\_id\_fkey FOREIGN KEY (group\_id) REFERENCES students\_groups(group\_id);-- ON DELETE CASCADE;

ALTER TABLE ONLY lessons

ADD CONSTRAINT lesson\_type\_check CHECK (lesson\_type ~\* '^Практика$' OR lesson\_type ~\* '^Теория$');

-- Индексы --------------------------------------------

CREATE INDEX ON lessons (student\_id);

-- Триггеры, процедуры и функции ----------------------

CREATE OR REPLACE FUNCTION compute\_certificate\_id()

RETURNS TRIGGER

LANGUAGE PLPGSQL AS

$$

BEGIN

NEW.graduate\_certificate\_id := '777-' ||

CURRENT\_DATE || '-' ||

LPAD(NEW.student\_id::text, 5, '0');

RETURN NEW;

END;

$$;

CREATE OR REPLACE TRIGGER set\_certificate\_id

BEFORE INSERT

ON graduates

FOR EACH ROW

EXECUTE FUNCTION compute\_certificate\_id();

------------

CREATE OR REPLACE PROCEDURE graduate\_student\_nodate(param\_student\_id int)

LANGUAGE SQL AS

$$

INSERT INTO graduates (student\_id

, graduation\_date)

VALUES (param\_student\_id, CURRENT\_DATE);

UPDATE students

SET car\_plate = DEFAULT

WHERE student\_id = param\_student\_id;

$$;

CREATE OR REPLACE PROCEDURE graduate\_student\_date(param\_student\_id int, param\_date date)

LANGUAGE SQL AS

$$

INSERT INTO graduates (student\_id

, graduation\_date)

VALUES (param\_student\_id, param\_date);

UPDATE students

SET car\_plate = DEFAULT

WHERE student\_id = param\_student\_id;

$$;

-----------------

-- empty group

CREATE OR REPLACE PROCEDURE add\_empty\_group\_id()

LANGUAGE PLPGSQL AS

$$

BEGIN

IF (SELECT count(\*) FROM students\_groups

WHERE group\_id = -1) = 0

THEN

INSERT INTO students\_groups(group\_id, enrollment\_date, graduation\_date)

VALUES (-1, '01-01-2023', '01-01-2023');

END IF;

END;

$$;

CALL add\_empty\_group\_id();

CREATE OR REPLACE FUNCTION dont\_remove\_empty\_group\_id\_func()

RETURNS TRIGGER

LANGUAGE PLPGSQL AS

$$

BEGIN

CALL add\_empty\_group\_id();

RETURN NEW;

END;

$$;

CREATE OR REPLACE TRIGGER dont\_remove\_empty\_group\_id

AFTER DELETE

ON students\_groups

FOR EACH STATEMENT

EXECUTE FUNCTION dont\_remove\_empty\_group\_id\_func();

--------------------

CREATE OR REPLACE FUNCTION no\_group\_id()

RETURNS TRIGGER

LANGUAGE PLPGSQL AS

$$

BEGIN

IF NEW.group\_id IS NULL

THEN NEW.group\_id = -1;

END IF;

RETURN NEW;

END;

$$;

CREATE OR REPLACE TRIGGER no\_group\_id\_given

BEFORE INSERT

ON students\_groups

FOR EACH ROW

EXECUTE FUNCTION no\_group\_id();

CREATE OR REPLACE TRIGGER no\_group\_id\_given

BEFORE INSERT

ON lessons

FOR EACH ROW

EXECUTE FUNCTION no\_group\_id();

CREATE OR REPLACE TRIGGER no\_group\_id\_given

BEFORE INSERT

ON students

FOR EACH ROW

EXECUTE FUNCTION no\_group\_id();

---------------------

CREATE OR REPLACE PROCEDURE add\_group\_date(param\_group\_id int, param\_enr\_date date)

LANGUAGE PLPGSQL AS

$$

BEGIN

IF param\_group\_id NOT IN (select group\_id from students\_groups)

THEN

INSERT INTO students\_groups (group\_id

, enrollment\_date)

VALUES (param\_group\_id, param\_enr\_date);

ELSE

raise notice 'That group (%) has been already added', param\_group\_id;

END IF;

END;

$$;

CREATE OR REPLACE PROCEDURE add\_group\_nodate(param\_group\_id int)

LANGUAGE PLPGSQL AS

$$

BEGIN

IF param\_group\_id NOT IN (select group\_id from students\_groups)

THEN

INSERT INTO students\_groups (group\_id

, enrollment\_date)

VALUES (param\_group\_id, CURRENT\_DATE);

ELSE

raise notice 'That group (%) has been already added', param\_group\_id;

END IF;

END;

$$;

-----------------------

CREATE OR REPLACE PROCEDURE add\_student(param\_ln varchar(30)

, param\_fn varchar(30)

, param\_pnmc varchar(30)

, param\_group\_id int

, param\_email varchar(30)

, param\_inn varchar(12))

LANGUAGE SQL AS

$$

INSERT INTO students(last\_name

, first\_name

, patronymic

, group\_id

, email

, inn)

VALUES (param\_ln, param\_fn, param\_pnmc, param\_group\_id, param\_email, param\_inn);

$$;

--.

# insert.sql

INSERT INTO school\_cars (car\_plate, brand, model, manufacture\_year)

VALUES

('О000ОО00', 'Без', 'Авто', 2022),

('А123АА77', 'Kia', 'Rio', 2019),

('Р456ОО77', 'Kia', 'Rio', 2020),

('Е789ММ77', 'Hyundai', 'Solaris', 2018),

('К123СС77', 'Hyundai', 'Solaris', 2019),

('Н456НН77', 'Volkswagen', 'Polo', 2020),

('О789РР77', 'Volkswagen', 'Polo', 2021),

('С123КС77', 'Skoda', 'Rapid', 2022),

('Т456МТ77', 'Skoda', 'Rapid', 2023),

('У789ЕР77', 'Daewoo', 'Matiz', 2016),

('А123НК77', 'Toyota', 'Corolla', 2020),

('К456КО77', 'Opel', 'Astra', 2019),

('Н789МК77', 'Renault', 'Sandero', 2021);

--select \* from school\_cars;

--delete from students\_groups;

--INSERT INTO students\_groups(group\_id) VALUES (1), (2), (3);

INSERT INTO students\_groups(group\_id, enrollment\_date, graduation\_date)

VALUES (1, '01-04-2025', '01-07-2025'),

(2, '22-04-2025', '22-07-2025'),

(3, '13-05-2025', '13-07-2025');

--select \* from students\_groups;

--delete from students

INSERT INTO students(last\_name, first\_name, patronymic, group\_id, email, inn)

VALUES

('Иванов', 'Иван', 'Александрович', 1, 'ivanov-ivan@mail.ru', '770000000001'),

('Петров', 'Сергей', 'Сергеевич', 1, 'petrov-sergei@gmail.com', '770000000002'),

('Смирнов', 'Алексей', 'Андреевич', 1, 'smirnov-aleksei@yandex.ru', '770000000003'),

('Попов', 'Михаил', 'Павлович', 1, 'popov-mikhail@rambler.ru', '770000000004'),

('Васильев', 'Дмитрий', 'Владимирович', 1, 'vasilev-dmitrii@bk.ru', '770000000005'),

('Федоров', 'Андрей', 'Константинович', 1, 'fedorov-andrei@mail.ru', '770000000006'),

('Сорокин', 'Евгений', 'Викторович', 1, 'sorokin-evgenii@yandex.ru', '770000000007'),

('Левин', 'Артем', 'Денисович', 1, 'levin-artem@gmail.com', '770000000008'),

('Николаев', 'Роман', 'Борисович', 1, 'nikolaev-roman@rambler.ru', '770000000009'),

('Семенов', 'Максим', 'Геннадьевич', 1, 'semenov-maksim@bk.ru', '770000000010'),

('Алексеев', 'Александр', 'Валерьевич', 2, 'alekseev-aleksandr@mail.ru', '770000000011'),

('Коваленко', 'Антон', 'Игоревич', 2, 'kovalenko-anton@yandex.ru', '770000000012'),

('Шестаков', 'Кирилл', 'Станиславович', 2, 'shestakov-kirill@rambler.ru', '770000000013'),

('Савельев', 'Владимир', 'Леонидович', 2, 'savelyev-vladimir@bk.ru', '770000000014'),

('Самохвалов', 'Степан', 'Дмитриевич', 2, 'samohvalov-stepan@mail.ru', '770000000015'),

('Chen', 'Yifeng', NULL, 2, 'chen-yifeng@gmail.com', '770000000016'), -- Китайские студент

('Lee', 'Seung-hyun', NULL, 2, 'lee-seunghyun@yahoo.co.kr', '770000000017'), -- Южнокорейский студент

('Sharma', 'Ankit', NULL, 2, 'sharma-ankit@hotmail.in', '770000000018'), -- Индийский студент

('Ishikawa', 'Naoki', NULL, 2, 'ishikawa-naoki@soramame.jp', '770000000019'), -- Японский студент

('Андреев', 'Денис', 'Игоревич', 2, 'andreev-denis@rambler.ru', '770000000020'),

('Петрова', 'Анна', 'Ивановна', 3, 'petrova-anna@mail.ru', '770000000021'),

('Смирнова', 'Ольга', 'Витальевна', 3, 'smirnova-olga@yandex.ru', '770000000022'),

('Попова', 'Марина', 'Петровна', 3, 'popova-marina@rambler.ru', '770000000023'),

('Васильева', 'Светлана', 'Георгиевна', 3, 'vasileva-svetlana@bk.ru', '770000000024'),

('Федорова', 'Дарья', 'Ростиславовна',3, 'fedorova-daria@mail.ru', '770000000025'),

('Сорокина', 'Инна', 'Анатольевна', 3, 'sorokina-inna@yandex.ru', '770000000026'),

('Львов', 'Арсен', 'Игнатьевич', 3, 'lvov-arsen@rambler.ru', '770000000027'),

('Морозов', 'Филипп', 'Леонидович', 3, 'morozov-filipp@bk.ru', '770000000028'),

('Черненко', 'Карина', 'Игоревна', 3, 'chernenko-karina@mail.ru', '770000000029'),

('Новиков', 'Эдуард', 'Владимирович', 3, 'novikov-eduard@yandex.ru', '770000000030');

--CALL graduate\_student\_nodate(1);

--select \* from students;

-- Выдача учащимся случайных учебных авто

DO

$$

DECLARE

rc record;

BEGIN

FOR rc IN SELECT student\_id FROM students

loop

raise notice '%', rc.student\_id;

UPDATE students s

SET car\_plate = (SELECT car\_plate FROM school\_cars ORDER BY random() LIMIT 1)

WHERE s.student\_id = rc.student\_id;

end loop;

END

$$;

------------------------------------------

INSERT INTO students(last\_name, first\_name, patronymic)

VALUES

('Петров', 'Сергей', 'Сергеевич'),

('Смирнов', 'Алексей', 'Андреевич'),

('Попов', 'Михаил', 'Павлович'),

('Васильев', 'Дмитрий', 'Владимирович'),

('Федоров', 'Андрей', 'Константинович'),

('Сорокин', 'Евгений', 'Викторович'),

('Левин', 'Артем', 'Денисович'),

('Николаев', 'Роман', 'Борисович'),

('Семенов', 'Максим', 'Геннадьевич'),

('Алексеев', 'Александр', 'Валерьевич'),

('Коваленко', 'Антон', 'Игоревич'),

('Шестаков', 'Кирилл', 'Станиславович'),

('Савельев', 'Владимир', 'Леонидович'),

('Самохвалов', 'Степан', 'Дмитриевич'),

('Wang', 'Linlin', NULL), -- китаец

('Patel', 'Sanjay', NULL), -- индус

('Tanaka', 'Ryosuke', NULL), -- японкц

('Nguyen', 'Thuy', NULL), -- вьетнамкц

('Petrovich', 'Marko', NULL); -- серб

DO

$$

DECLARE

rc record;

BEGIN

FOR rc IN SELECT student\_id FROM students

loop

IF rc.student\_id BETWEEN 31 AND 49

THEN CALL graduate\_student\_nodate(rc.student\_id);

END IF;

end loop;

END

$$;

--select \* from graduates;

INSERT INTO professors(professor\_id, last\_name, first\_name, patronymic, experience\_years)

VALUES

(1, 'Иванов', 'Иван', 'Александрович', 15),

(2, 'Петров', 'Сергей', 'Сергеевич', 20),

(3, 'Саркисян', 'Гарегин', 'Арташесович', 18),

(4, 'Давтян', 'Аваг', 'Варданович', 25),

(5, 'Маркосян', 'Виген', 'Албертович', 12),

(6, 'Федоров', 'Андрей', 'Константинович', 30),

(7, 'Сорокин', 'Евгений', 'Викторович', 10),

(8, 'Мелконян', 'Армен', 'Айказович', 22),

(9, 'Арутюнян', 'Арам', 'Артакович', 14),

(10, 'Николаев', 'Роман', 'Борисович', 28),

(11, 'Алескерян', 'Акоп', 'Агабекович', 35),

(12, 'Коваленко', 'Антон', 'Игоревич', 20),

(13, 'Шестаков', 'Кирилл', 'Станиславович', 16),

(14, 'Савельев', 'Владимир', 'Леонидович', 32),

(15, 'Самохвалов', 'Степан', 'Дмитриевич', 19),

(16, 'Львов', 'Арсен', 'Игнатьевич', 24),

(17, 'Морозов', 'Филипп', 'Леонидович', 27),

(18, 'Черненко', 'Карина', 'Игоревна', 30),

(19, 'Новиков', 'Эдуард', 'Владимирович', 38),

(20, 'Петрова', 'Анна', 'Ивановна', 17);

--select \* from professors;

INSERT INTO employments(professor\_id, employed\_since, salary)

VALUES

(1, '2023-01-15', '75000'),

(2, '2023-03-20', '100000'),

(3, '2023-05-10', '80000'),

(4, '2023-07-01', '120000'),

(5, '2023-09-15', '65000'),

(6, '2023-11-20', '95000'),

(7, '2024-01-10', '60000'),

(8, '2024-03-01', '110000'),

(9, '2024-05-15', '70000'),

(10, '2024-07-20', '130000'),

(11, '2024-09-10', '150000'),

(12, '2024-11-01', '85000'),

(13, '2023-02-15', '75000'),

(14, '2023-04-20', '90000'),

(15, '2023-06-10', '65000'),

(16, '2023-08-01', '80000'),

(17, '2023-10-15', '105000'),

(18, '2023-12-20', '60000'),

(19, '2024-02-10', '125000'),

(20, '2024-04-01', '55000');

--select \* from employments;

--delete from lessons where 1=1;

--truncate lessons restart identity;

--select \* from lessons

INSERT INTO lessons(lesson\_type, professor\_id, student\_id, group\_id, lesson\_date)

VALUES

('Теория', 6, 1, 1, '13-05-2025'),

('Теория', 6, 2, 1, '13-05-2025'),

('Теория', 6, 3, 1, '13-05-2025'),

('Теория', 6, 4, 1, '13-05-2025'),

('Теория', 6, 5, 1, '13-05-2025'),

('Теория', 6, 6, 1, '13-05-2025'),

('Теория', 6, 7, 1, '13-05-2025'),

('Теория', 6, 8, 1, '13-05-2025'),

('Теория', 6, 9, 1, '13-05-2025'),

('Теория', 6, 10, 1, '13-05-2025'),

('Практика', 19, 5, NULL, '03-04-2025'),

('Практика', 18, 12, NULL, '25-04-2025'),

('Практика', 10, 28, NULL, '17-05-2025'),

('Практика', 19, 4, NULL, '04-04-2025'),

('Практика', 17, 20, NULL, '26-04-2025'),

('Практика', 20, 28, NULL, '18-05-2025'),

('Практика', 13, 2, NULL, '05-04-2025'),

('Практика', 18, 20, NULL, '27-04-2025'),

('Практика', 3, 21, NULL, '19-05-2025'),

('Практика', 14, 2, NULL, '06-04-2025'),

('Практика', 12, 20, NULL, '28-04-2025'),

('Практика', 20, 23, NULL, '20-05-2025'),

('Практика', 5, 3, NULL, '07-04-2025'),

('Практика', 15, 17, NULL, '29-04-2025'),

('Практика', 4, 25, NULL, '21-05-2025'),

('Практика', 3, 3, NULL, '08-04-2025'),

('Практика', 13, 15, NULL, '30-04-2025'),

('Практика', 7, 28, NULL, '22-05-2025'),

('Практика', 5, 3, NULL, '09-04-2025'),

('Практика', 12, 11, NULL, '01-05-2025'),

('Практика', 11, 29, NULL, '23-05-2025'),

('Практика', 11, 8, NULL, '10-04-2025'),

('Практика', 11, 18, NULL, '02-05-2025'),

('Практика', 19, 27, NULL, '24-05-2025'),

('Практика', 1, 4, NULL, '11-04-2025'),

('Практика', 3, 18, NULL, '03-05-2025'),

('Практика', 3, 26, NULL, '25-05-2025'),

('Практика', 4, 3, NULL, '12-04-2025'),

('Практика', 1, 15, NULL, '04-05-2025'),

('Практика', 18, 27, NULL, '26-05-2025'),

('Практика', 6, 4, NULL, '13-04-2025'),

('Практика', 7, 13, NULL, '05-05-2025'),

('Практика', 15, 30, NULL, '27-05-2025'),

('Практика', 13, 3, NULL, '14-04-2025'),

('Практика', 11, 15, NULL, '06-05-2025'),

('Практика', 12, 23, NULL, '28-05-2025'),

('Практика', 10, 10, NULL, '15-04-2025'),

('Практика', 18, 14, NULL, '07-05-2025'),

('Практика', 15, 26, NULL, '29-05-2025'),

('Практика', 17, 9, NULL, '16-04-2025'),

('Практика', 16, 15, NULL, '08-05-2025'),

('Практика', 8, 26, NULL, '30-05-2025'),

('Практика', 10, 9, NULL, '17-04-2025'),

('Практика', 2, 12, NULL, '09-05-2025'),

('Практика', 7, 22, NULL, '31-05-2025'),

('Практика', 15, 8, NULL, '18-04-2025'),

('Практика', 20, 13, NULL, '10-05-2025'),

('Практика', 20, 30, NULL, '01-06-2025'),

('Практика', 18, 8, NULL, '19-04-2025'),

('Практика', 8, 17, NULL, '11-05-2025'),

('Практика', 14, 27, NULL, '02-06-2025'),

('Практика', 4, 8, NULL, '20-04-2025'),

('Практика', 2, 14, NULL, '12-05-2025'),

('Практика', 10, 27, NULL, '03-06-2025'),

('Практика', 7, 6, NULL, '21-04-2025'),

('Практика', 20, 12, NULL, '13-05-2025'),

('Практика', 18, 21, NULL, '04-06-2025'),

('Практика', 5, 9, NULL, '22-04-2025'),

('Практика', 20, 14, NULL, '14-05-2025'),

('Практика', 19, 28, NULL, '05-06-2025'),

('Теория', 10, 21, 3, '24-06-2025'),

('Теория', 10, 22, 3, '24-06-2025'),

('Теория', 10, 23, 3, '24-06-2025'),

('Теория', 10, 24, 3, '24-06-2025'),

('Теория', 10, 25, 3, '24-06-2025'),

('Теория', 10, 26, 3, '24-06-2025'),

('Теория', 10, 27, 3, '24-06-2025'),

('Теория', 10, 28, 3, '24-06-2025'),

('Теория', 10, 29, 3, '24-06-2025'),

('Теория', 10, 30, 3, '24-06-2025'),

('Практика', 2, 7, NULL, '24-04-2025'),

('Практика', 13, 13, NULL, '16-05-2025'),

('Практика', 15, 30, NULL, '07-06-2025'),

('Практика', 16, 2, NULL, '25-04-2025'),

('Практика', 12, 16, NULL, '17-05-2025'),

('Практика', 7, 26, NULL, '08-06-2025'),

('Практика', 19, 7, NULL, '26-04-2025'),

('Практика', 12, 14, NULL, '18-05-2025'),

('Практика', 3, 26, NULL, '09-06-2025'),

('Практика', 7, 6, NULL, '27-04-2025'),

('Практика', 10, 17, NULL, '19-05-2025'),

('Практика', 13, 28, NULL, '10-06-2025'),

('Практика', 5, 7, NULL, '28-04-2025'),

('Практика', 17, 15, NULL, '20-05-2025'),

('Практика', 15, 21, NULL, '11-06-2025'),

('Практика', 12, 8, NULL, '29-04-2025'),

('Практика', 10, 19, NULL, '21-05-2025'),

('Практика', 18, 28, NULL, '12-06-2025'),

('Практика', 2, 9, NULL, '30-04-2025'),

('Практика', 13, 11, NULL, '22-05-2025'),

('Практика', 10, 30, NULL, '13-06-2025'),

('Практика', 10, 3, NULL, '01-05-2025'),

('Практика', 10, 15, NULL, '23-05-2025'),

('Практика', 19, 26, NULL, '14-06-2025'),

('Практика', 15, 2, NULL, '02-05-2025'),

('Практика', 4, 17, NULL, '24-05-2025'),

('Практика', 16, 29, NULL, '15-06-2025'),

('Практика', 8, 2, NULL, '03-05-2025'),

('Практика', 14, 18, NULL, '25-05-2025'),

('Практика', 9, 28, NULL, '16-06-2025'),

('Практика', 19, 7, NULL, '04-05-2025'),

('Практика', 1, 18, NULL, '26-05-2025'),

('Практика', 20, 30, NULL, '17-06-2025'),

('Практика', 3, 2, NULL, '05-05-2025'),

('Практика', 6, 20, NULL, '27-05-2025'),

('Практика', 7, 22, NULL, '18-06-2025'),

('Практика', 2, 7, NULL, '06-05-2025'),

('Практика', 16, 19, NULL, '28-05-2025'),

('Практика', 8, 23, NULL, '19-06-2025'),

('Практика', 6, 4, NULL, '07-05-2025'),

('Практика', 12, 17, NULL, '29-05-2025'),

('Практика', 13, 28, NULL, '20-06-2025'),

('Практика', 13, 7, NULL, '08-05-2025'),

('Практика', 8, 20, NULL, '30-05-2025'),

('Практика', 10, 28, NULL, '21-06-2025'),

('Практика', 15, 2, NULL, '09-05-2025'),

('Практика', 15, 16, NULL, '31-05-2025'),

('Практика', 19, 30, NULL, '22-06-2025'),

('Практика', 7, 3, NULL, '10-05-2025'),

('Практика', 4, 17, NULL, '01-06-2025'),

('Практика', 11, 22, NULL, '23-06-2025'),

('Практика', 7, 4, NULL, '11-05-2025'),

('Практика', 13, 11, NULL, '02-06-2025'),

('Практика', 13, 23, NULL, '24-06-2025'),

('Практика', 17, 3, NULL, '12-05-2025'),

('Практика', 20, 17, NULL, '03-06-2025'),

('Практика', 11, 29, NULL, '25-06-2025'),

('Практика', 16, 6, NULL, '13-05-2025'),

('Практика', 7, 15, NULL, '04-06-2025'),

('Практика', 17, 30, NULL, '26-06-2025'),

('Теория', 3, 11, 2, '03-06-2025'),

('Теория', 3, 12, 2, '03-06-2025'),

('Теория', 3, 13, 2, '03-06-2025'),

('Теория', 3, 14, 2, '03-06-2025'),

('Теория', 3, 15, 2, '03-06-2025'),

('Теория', 3, 16, 2, '03-06-2025'),

('Теория', 3, 17, 2, '03-06-2025'),

('Теория', 3, 18, 2, '03-06-2025'),

('Теория', 3, 19, 2, '03-06-2025'),

('Теория', 3, 20, 2, '03-06-2025'),

('Практика', 17, 3, NULL, '15-05-2025'),

('Практика', 19, 13, NULL, '06-06-2025'),

('Практика', 3, 27, NULL, '28-06-2025'),

('Практика', 9, 6, NULL, '16-05-2025'),

('Практика', 5, 19, NULL, '07-06-2025'),

('Практика', 20, 26, NULL, '29-06-2025'),

('Практика', 8, 10, NULL, '17-05-2025'),

('Практика', 8, 19, NULL, '08-06-2025'),

('Практика', 3, 22, NULL, '30-06-2025'),

('Практика', 2, 4, NULL, '18-05-2025'),

('Практика', 1, 14, NULL, '09-06-2025'),

('Практика', 15, 26, NULL, '01-07-2025'),

('Практика', 20, 6, NULL, '19-05-2025'),

('Практика', 20, 13, NULL, '10-06-2025'),

('Практика', 4, 25, NULL, '02-07-2025'),

('Практика', 2, 4, NULL, '20-05-2025'),

('Практика', 9, 13, NULL, '11-06-2025'),

('Практика', 5, 30, NULL, '03-07-2025'),

('Практика', 12, 6, NULL, '21-05-2025'),

('Практика', 6, 11, NULL, '12-06-2025'),

('Практика', 11, 24, NULL, '04-07-2025'),

('Практика', 8, 6, NULL, '22-05-2025'),

('Практика', 4, 17, NULL, '13-06-2025'),

('Практика', 7, 28, NULL, '05-07-2025'),

('Практика', 6, 1, NULL, '23-05-2025'),

('Практика', 15, 13, NULL, '14-06-2025'),

('Практика', 15, 21, NULL, '06-07-2025'),

('Практика', 12, 3, NULL, '24-05-2025'),

('Практика', 6, 11, NULL, '15-06-2025'),

('Практика', 3, 29, NULL, '07-07-2025'),

('Практика', 18, 8, NULL, '25-05-2025'),

('Практика', 5, 17, NULL, '16-06-2025'),

('Практика', 15, 30, NULL, '08-07-2025'),

('Практика', 20, 3, NULL, '26-05-2025'),

('Практика', 19, 14, NULL, '17-06-2025'),

('Практика', 16, 29, NULL, '09-07-2025'),

('Практика', 16, 4, NULL, '27-05-2025'),

('Практика', 10, 14, NULL, '18-06-2025'),

('Практика', 14, 29, NULL, '10-07-2025'),

('Практика', 16, 9, NULL, '28-05-2025'),

('Практика', 13, 12, NULL, '19-06-2025'),

('Практика', 6, 29, NULL, '11-07-2025'),

('Практика', 13, 6, NULL, '29-05-2025'),

('Практика', 8, 18, NULL, '20-06-2025'),

('Практика', 17, 28, NULL, '12-07-2025'),

('Практика', 10, 7, NULL, '30-05-2025'),

('Практика', 6, 18, NULL, '21-06-2025'),

('Практика', 4, 29, NULL, '13-07-2025'),

('Практика', 8, 9, NULL, '31-05-2025'),

('Практика', 2, 19, NULL, '22-06-2025'),

('Практика', 17, 29, NULL, '14-07-2025'),

('Практика', 10, 10, NULL, '01-06-2025'),

('Практика', 3, 20, NULL, '23-06-2025'),

('Практика', 10, 22, NULL, '15-07-2025'),

('Практика', 4, 6, NULL, '02-06-2025'),

('Практика', 16, 16, NULL, '24-06-2025'),

('Практика', 3, 25, NULL, '16-07-2025'),

('Практика', 1, 6, NULL, '03-06-2025'),

('Практика', 2, 16, NULL, '25-06-2025'),

('Практика', 12, 24, NULL, '17-07-2025');

--.

# queries.sql

SELECT \* FROM employments;

SELECT \* FROM graduates;

SELECT \* FROM lessons;

SELECT \* FROM professors;

SELECT \* FROM school\_cars;

SELECT \* FROM students;

SELECT \* FROM students\_groups;

/\*

№1

Выбрать ФИО всех студенток из всех нечетных групп, зачисленных после 31.12.2024

\*/

SELECT last\_name, first\_name, patronymic --, enrollment\_date

FROM students s

JOIN students\_groups sg ON s.group\_id = sg.group\_id

WHERE last\_name ~\* '.\*а$'

AND s.group\_id % 2 = 1

AND enrollment\_date > '31-12-2024';

/\*

№2

Вывести всех студентов, чья почта не зарегистрированна на @gmail.com или mail.ru.

Отсортировать по дате поступления (сначала новые, потом уже те, что раньше поступили),

ФИО (По Возрастающей)

\*/

SELECT last\_name, first\_name, patronymic, enrollment\_date

FROM students s

JOIN students\_groups sg ON s.group\_id = sg.group\_id

EXCEPT

SELECT last\_name, first\_name, patronymic, enrollment\_date

FROM students s

JOIN students\_groups sg ON s.group\_id = sg.group\_id

WHERE email ~\* '.\*@gmail.com'

OR email ~\* '.\*@mail.ru'

ORDER BY 4 DESC, 1, 2, 3;

/\*

№3

Вывести всех преподавателей с окладом >100.000. Отсортировать по Окладу по убывающей

\*/

SELECT last\_name, first\_name, patronymic, salary

FROM professors p

JOIN employments emp ON emp.professor\_id = p.professor\_id

WHERE salary > 100000::money

ORDER BY 4 DESC;

/\*

№4

Вывести количество преподавателей к каждой "вилке" окладов с шагом 30000 рублей

\*/

SELECT min\_salary, max\_salary, count(emp.\*)

FROM employments emp

RIGHT OUTER JOIN -- Чтобы даже если в данной "вилке" нету людей с тако зп - выводим диапазон

( VALUES (000000::money, 029999.99::money),

(030000::money, 059999.99::money),

(060000::money, 089999.99::money),

(090000::money, 119999.99::money),

(120000::money, 149999.99::money),

(150000::money, 179999.99::money)

) AS s (min\_salary, max\_salary)

ON emp.salary BETWEEN s.min\_salary AND s.max\_salary

GROUP BY s.min\_salary, s.max\_salary

ORDER BY s.min\_salary ASC;

/\*

№5

Найти среднюю ЗП преподавателей НЕ УЧИТЫВАЯ 2 самые высокие и 2 самые низкие

\*/

SELECT avg(nt.filtered\_salary::numeric)::money AS "Средняя отфильтрованная ЗП"

FROM (

SELECT emp.professor\_id, emp.salary FROM employments emp

EXCEPT (

(SELECT emp1.professor\_id, emp1.salary FROM employments emp1

ORDER BY 1 ASC

LIMIT 2)

UNION

(SELECT emp2.professor\_id, emp2.salary from employments emp2

ORDER BY 1 DESC

LIMIT 2)

)

) AS nt (filtered\_professor\_id, filtered\_salary);

/\*

№6

Вывести студентов, у которых ВНУТРИ имени почты содержится латинская букву A,

отсортировать по фамилиям в обратном порядке.

\*/

SELECT last\_name, first\_name, patronymic, email

FROM students s

WHERE email ~\* '.\*a.\*@.\*'

ORDER BY 1 DESC;

/\*

№7

Вывести всех преподавателей, у которых свыше 15 лет стажа

и при этом они трудоустроены до лета 2024 года.

\*/

SELECT last\_name, first\_name, patronymic,

experience\_years, employed\_since

FROM professors p

LEFT JOIN employments emp ON emp.professor\_id = p.professor\_id

WHERE p.experience\_years > 15

AND emp.employed\_since < '01-06-2024'::date

ORDER BY 4 DESC, 5, 1, 2, 3;

/\*

№8

Вывести все групповые занятия (без повторений) с датой зачисления группы.

\*/

SELECT DISTINCT lesson\_id, lesson\_type, professor\_id, sg.group\_id, enrollment\_date

FROM lessons l

INNER JOIN students\_groups sg ON sg.group\_id = l.group\_id;

/\*

№9

Вывести всех преподавателей с ЗП выше средней по Автошколе.

Первой строкой также вывести среднюю ЗП.

\*/

select last\_name, first\_name, patronymic, salary

from (

(

SELECT last\_name, first\_name, patronymic, salary, t

FROM professors p

JOIN employments e ON p.professor\_id = e.professor\_id

CROSS JOIN (VALUES (2)) as sid (t)

WHERE e.salary::numeric > (SELECT AVG(salary::numeric) FROM employments)

)

UNION

(

SELECT last\_name, first\_name, patronymic, temp\_emp.salary, t

FROM (

SELECT AVG(emp.salary::numeric)::money AS "salary" FROM employments emp

) as temp\_emp (salary)

CROSS JOIN (VALUES (1)) as sid (t)

CROSS JOIN (VALUES ('-- Средняя'::varchar(30), 'Заработная'::varchar(30), 'Плата ='::varchar(30)))

as f (last\_name, first\_name, patronymic)

)

) as foo

ORDER BY t, salary ASC;

/\*

№10

Вывести, сколько и каких занятий провел каждый преподаватель

\*/

SELECT DISTINCT p.professor\_id, p.last\_name, p.first\_name, p.patronymic, lesson\_type, count(lesson\_date) AS "Число занятий"

FROM professors p

CROSS JOIN lessons l

WHERE p.professor\_id = l.professor\_id

GROUP BY 1,2,3,4,5

ORDER BY lesson\_type DESC, 1 ASC;

/\*

№11

Студентов, прошедших как теорию, так и практику

\*/

SELECT last\_name, first\_name, patronymic

FROM lessons l

LEFT JOIN students s ON s.student\_id = l.student\_id

GROUP BY 1,2,3

HAVING COUNT(DISTINCT lesson\_type) = 2;

/\*

№12

Всех выпускников, что не являются преподавателями в нашей Автошколе

\*/

SELECT g.last\_name, g.first\_name, g.patronymic

FROM graduates g

EXCEPT --INTERSECT

SELECT p.last\_name, p.first\_name, p.patronymic

FROM professors p

ORDER BY 1,2,3;

/\*

№13

Студентов, которым преподавал один и тот же профессор и теорию, и практику

\*/

SELECT s.last\_name, s.first\_name, s.patronymic, professor\_id

FROM lessons l

LEFT JOIN students s ON s.student\_id = l.student\_id

GROUP BY 1,2,3,4

HAVING count(distinct lesson\_type) = 2;

/\*

№14

Топ 3 самых частых учебных авто у студентов

\*/

SELECT sc.brand, sc.model, COUNT(s.student\_id) AS student\_count

FROM school\_cars sc

JOIN students s ON sc.car\_plate = s.car\_plate

GROUP BY sc.brand, sc.model

ORDER BY student\_count DESC

LIMIT 3;

/\*

№15

Вывести преподавателей, у которых было больше 5 практических занятий за май 2025 года

\*/

SELECT p.professor\_id, p.last\_name, COUNT(DISTINCT l.lesson\_date) --, lesson\_date

FROM professors p

JOIN lessons l ON p.professor\_id = l.professor\_id

WHERE l.lesson\_date BETWEEN '2025-05-01' AND '2025-05-31'

AND l.lesson\_type ~\* 'практика'

GROUP BY p.professor\_id, p.last\_name

HAVING COUNT(DISTINCT l.lesson\_date) > 5;

/\*

№16

Вывести 3 студентов, у которых было большего всего практических занятий

и 3 с наименьшим их числом

\*/

SELECT \* FROM (

(

SELECT s.last\_name, s.first\_name, s.patronymic, count(\*)

FROM lessons l

LEFT JOIN students s ON s.student\_id = l.student\_id

WHERE lesson\_type ~\* 'практика'

GROUP BY 1,2,3

ORDER BY count DESC

LIMIT 3

)

UNION

(

SELECT s.last\_name, s.first\_name, s.patronymic, count(\*)

FROM lessons l

LEFT JOIN students s ON s.student\_id = l.student\_id

WHERE lesson\_type ~\* 'практика'

GROUP BY 1,2,3

ORDER BY count ASC

LIMIT 3

)

) as foo

ORDER BY count DESC;

/\*

№17

Вывести все группы, в которых есть иностранные студенты (нету отчеств)

\*/

SELECT DISTINCT group\_id FROM students

WHERE patronymic IS NULL;

/\*

№18

Вывести дату последнего практического занятия у каждого студента и количество этих занятий

\*/

SELECT s.last\_name, s.first\_name, s.patronymic,

MAX(l.lesson\_date) AS "Дата последнего занятия",

COUNT(\*) AS "Число занятий"

FROM students s

JOIN lessons l ON s.student\_id = l.student\_id

WHERE lesson\_type ~\* 'практика'

GROUP BY s.student\_id

ORDER BY 1,2,3;

/\*

№19

Вывести id студентов, что занимались с одним и тем же преподавателем 2 или более раз

\*/

SELECT student\_id, professor\_id, COUNT(lesson\_date)

FROM lessons

GROUP BY student\_id, professor\_id

HAVING COUNT(lesson\_date) >= 2;

/\*

№20

Вывести студентов, у которых занятия вёл хотя бы 1 из 3 преподавателей

с самой большой ЗП

\*/

SELECT s.last\_name, s.first\_name, s.patronymic

FROM students s

WHERE s.student\_id IN (

SELECT DISTINCT l.student\_id FROM lessons l

JOIN employments emp ON emp.professor\_id = l.professor\_id

WHERE l.student\_id = s.student\_id

AND emp.salary IN (

SELECT salary FROM employments

ORDER BY salary DESC LIMIT 3

)

);