МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Государственное бюджетное образовательное учреждение Российской Федерации

Национальный исследовательский университет «Московский институт электронной техники»

Институт микроприборов и систем управления имени Л.Н. Преснухина

Проектная работа

Выполнили студенты:

Арюхин Александр  
Фролов Георгий

группа ИВТ-42

Научный руководитель:

Балабаев Сергей Андреевич

«30» декабря 2023 г.

Зеленоград, 2023

1. **Подробное описание структуры базы данных.**

База данных состоит из 6 таблиц:

1. scooter. Содержит:

**scooter\_id** integer NOT NULL, - уникальный идентификатор каждого самоката(PK).

**parking\_id** integer, - уникальный идентификатор места стоянки, на котором он сейчас находится.(FK).

**warehouse\_id** integer, - уникальный идентификатор склада, к которому привязан данный самокат(FK).

**model\_**id integer NOT NULL, - уникальный идентификатор модели самоката(FK).

**latitude** numeric(13,10), - текущая ширина самоката(координатная).

**longitude** numeric(13,10), - текущая долгота самоката(координатная).

**charge\_lvl** numeric(5,2), - уровень заряда аккумулятора самоката.

**battery\_wear** numeric(3,2), - состояние износа аккумулятора, где 1 -абсолютно новый.

**state** character varying(30) – состояние самоката(может быть в использовании, в ожидании, разряжен, на ремонте).

1. model. Содержит:

**model\_id** integer NOT NULL, - уникальный идентификатор модели самоката(PK).

**color** character varying(30) COLLATE pg\_catalog."default" NOT NULL, - цвет самоката .

**manufacturer\_country** character varying(30) COLLATE pg\_catalog."default" NOT NULL, - страна производства модели самоката.

**manufacturer**\_email character varying(30) COLLATE pg\_catalog."default", - почта производителя данной модели, для связи в случае подписания дополнительных контрактов или решения важных вопросов.

**warranty\_period** integer, - гарантийный срок данной модели.

1. parking. Содержит:

**parking\_id** integer NOT NULL, - уникальный идентификатор места стоянки (PK)

**warehouse\_id** integer NOT NULL, уникальный идентификатор склада, к которому привязана эта стоянка (FK).

**parking\_name** character varying(50) COLLATE pg\_catalog."default" NOT NULL, - название стоянки.

**number\_places** integer NOT NULL, - информация о максимальном количестве самокатов на этой стоянке.

**note** character varying(200) COLLATE pg\_catalog."default", - примечания об этой стоянке, рядом с чем находится, как проще найти.

1. warehouse. Содержит:

**warehouse\_id** integer NOT NULL, - уникальный идентификатор склада(PK).

**number\_personnel** integer NOT NULL, - количество персонала, требующегося для обслуживания склада.

**number\_places** integer NOT NULL, - максимальное количество мест для самокатов на этом складе(вместимость).

**service\_cost** money, - стоимость обслуживания склада.

1. info\_employees. Содержит:

**employees\_id** integer NOT NULL, - уникальный идентификатор сотрудника(PK).

**surname** character varying(30) COLLATE pg\_catalog."default" NOT NULL, - фамилия сотрудника.

**name** character varying(30) COLLATE pg\_catalog."default" NOT NULL, - имя сотрудника.

**patronymic** character varying(30) COLLATE pg\_catalog."default", - отчество сотрудника.

**birthdate** date NOT NULL, - дата рождения сотрудника.

**experience** integer, - стаж работы.

**post** character varying(200) COLLATE pg\_catalog."default" NOT NULL, - должность.

**criminal\_record** character varying(30) COLLATE pg\_catalog."default", - судимость(для реализации аккредитованной государственной программы по трудоустройству после отбытия наказания).

**salary** money NOT NULL, - заробатная плата.

1. employees. Содержит:

**employees\_id** integer NOT NULL, - уникальный идентификатор сотрудника(PK).

**warehouse\_id** integer NOT NULL, - уникальный идентификатор склада, на котором он работает(PK).

**contract**\_number integer, - номер контракта сотрудника.

**wage\_rate** integer, - ставка сотрудника.

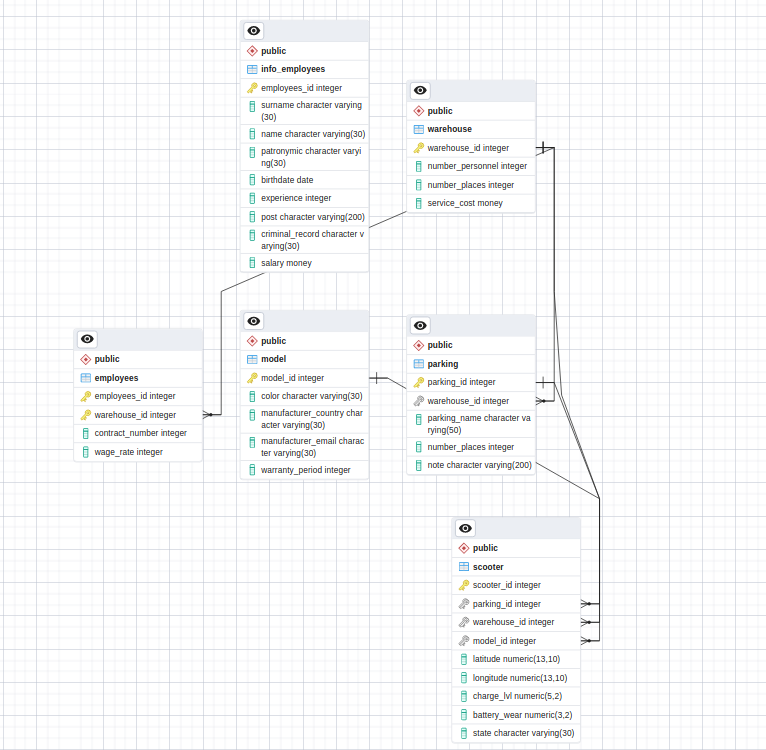
В городе, где ритм жизни быстрый, и заботы о экологии становятся всё более важными, компания по прокату электросамокатов решает внедрить современную базу данных учёта. Эта база данных играет ключевую роль в управлении и поддержании парка электросамокатов.

Через эту систему, операторы могут легко отслеживать распределение самокатов по городу, контролировать их техническое состояние и обслуживание. Каждый электросамокат оборудован умными датчиками, передающими данные в базу данных: уровень заряда батареи, износ компонентов, их местоположение и статус.

База данных позволяет оптимизировать ресурсы, следить за тем, где больше всего спрос на электросамокаты, и распределять их по популярным районам города. Также она предоставляет операторам возможность оперативно реагировать на технические проблемы, например, если износ батареи приближается к критическому уровню, или если самокат выходит из строя.

Благодаря базе данных, компания по прокату электросамокатов не только повышает эффективность своей деятельности, но и вносит вклад в создание более устойчивой и экологически чистой городской среды.

1. **Схема базы данных**



1. **Список созданных запросов с их описанием**

Получить список всех самокатов:

SELECT \* FROM public.scooter

ORDER BY scooter\_id ASC

Получить список всех моделей:

SELECT \* FROM public.model

ORDER BY model\_id ASC

Получить список всех складов:

SELECT \* FROM public.warehouse

ORDER BY warehouse\_id ASC

Получить список всех мест стоянок:

SELECT \* FROM public.parking

ORDER BY parking\_id ASC

Получить список всех Сотрудников:

SELECT \* FROM public.employees

ORDER BY employees\_id ASC, warehouse\_id ASC

Получить информацию о сотрудниках, которые работают больше нормы:

SELECT \* FROM employees WHERE wage\_rate > 1;

Получить информацию о сломанных самокатах:

SELECT \* FROM scooter WHERE state = 'faulty';

Получить информацию об импортозамещении (самокаты, сделанные в России):

SELECT \* FROM model WHERE manufacturer\_country = 'Russia';

Получить информацию об самокатах, у которых уровень износа батареи > 50:

SELECT \* FROM scooter WHERE battery\_wear < 0.5;

Получить информацию о складах, стоимость обслуживания которых больше 250 000руб:

SELECT \* FROM warehouse WHERE service\_cost > money(250000);

Получить средний стаж сотрудников:

SELECT AVG(experience) AS средний\_стаж

FROM info\_employees;

Получить места больших стоянок самокатов(могут стоять больше 10)

SELECT \* FROM parking WHERE number\_places > 10;

Получить данные всех доставщиков самокатов(полиция обратилась, ищут закладчика)

SELECT \* FROM info\_employees WHERE post = 'Доставщик';

Получить количество рабочих с судимостью(по программе трудоустройства после заключения)

SELECT COUNT(employees\_id) FROM info\_employees WHERE criminal\_record = 'Да';

Получить количество самокатов каждой модели:

SELECT model.model\_id, COUNT(scooter.scooter\_id) AS количество\_самокатов

FROM model

JOIN scooter ON model.model\_id = scooter.model\_id

GROUP BY model.model\_id

Получить количество работников, которым в следующем году выходить на пенсию (выход на пенсию в 63 года)

SELECT \* FROM info\_employees

WHERE EXTRACT(YEAR FROM birthdate) = 1961;

Показать самокаты, доступные для аренды

SELECT scooter\_id FROM scooter WHERE state = 'waiting';

Найти аккумуляторы с уровнем заряда равным 100%

SELECT scooter\_id FROM scooter WHERE charge\_lvl = 100;

Вывести все самокаты желтых цветов(длядальнейшей перекраски в связи с событиями)

SELECT scooter\_id

FROM scooter

INNER JOIN model ON scooter.model\_id = model.model\_id

WHERE model.color = 'Yellow';

Вывести все самокаты, у которых гарантийный срок больше 1 года:

SELECT scooter\_id

FROM scooter

INNER JOIN model ON scooter.model\_id = model.model\_id

WHERE warranty\_period > 1;

1. **Список созданных процедур, функций, представлений, триггеров с их описанием, параметрами работы**

Создание триггера, который бы менял состояние state на fault если battery\_wear становится ниже 0.5. Этот триггер нужен для отслеживания самокатов, у которых уровень износа аккумулятора становится более 50%.

Этот код создает функцию update\_state\_on\_battery\_wear, которая является телом триггера. Функция выполняет проверку: если battery\_wear новой записи меньше 0.5, то изменяет значение state на 'fault'. Эта функция возвращает измененную запись.

CREATE OR REPLACE FUNCTION update\_state\_on\_battery\_wear()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.battery\_wear < 0.5 THEN

NEW.state = 'fault';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

В этой части кода создается сам триггер update\_state\_trigger. Триггер срабатывает перед обновлением записи в таблице scooter. Каждый раз, когда происходит попытка обновить запись, триггер вызывает функцию update\_state\_on\_battery\_wear.

CREATE TRIGGER update\_state\_trigger

BEFORE UPDATE ON scooter

FOR EACH ROW

EXECUTE FUNCTION update\_state\_on\_battery\_wear();

1. **Рекомендации по работе с базой данных – оптимизация с помощью индексов**

Также, для оптимизации базы данных были добавлены индексы.

Индекс – специальная структура данных, которая связана с таблицей и создается на основе данных, содержащихся в ней. Основная цель создания индексов – повышение производительности функционирования базы данных.

Таким образом, при больших объемах базы данных, запросы, содержащие ключевое слово WHERE будут выполняться быстрее. Таблица, которая содержит и всегда будет содержать наибольшее количество данных – scooter, поэтому мы добавили индексы в эту таблицу по столбцу scooter\_id:

CREATE INDEX id\_index ON scooter (scooter\_id);

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

Код для создания базы даных:

-- Table: public.scooter

-- DROP TABLE IF EXISTS public.scooter;

CREATE TABLE IF NOT EXISTS public.scooter

(

scooter\_id integer NOT NULL,

parking\_id integer,

warehouse\_id integer,

model\_id integer NOT NULL,

latitude numeric(13,10),

longitude numeric(13,10),

charge\_lvl numeric(5,2),

battery\_wear numeric(3,2),

state character varying(30) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT scooter\_pkey PRIMARY KEY (scooter\_id),

CONSTRAINT scooter\_model\_key FOREIGN KEY (model\_id)

REFERENCES public.model (model\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT scooter\_parking\_key FOREIGN KEY (parking\_id)

REFERENCES public.parking (parking\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT scooter\_warehouse\_key FOREIGN KEY (warehouse\_id)

REFERENCES public.warehouse (warehouse\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT scooter\_charge\_lvl\_check CHECK (charge\_lvl >= 0::numeric AND charge\_lvl <= 100::numeric),

CONSTRAINT scooter\_battery\_wear\_check CHECK (battery\_wear >= 0::numeric AND battery\_wear <= 1::numeric),

CONSTRAINT scooter\_state\_check CHECK (state::text = 'is used'::text OR state::text = 'waiting'::text OR state::text = 'discharged'::text OR state::text = 'faulty'::text)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.scooter

OWNER to postgres;

-- Index: id\_index

-- DROP INDEX IF EXISTS public.id\_index;

CREATE INDEX IF NOT EXISTS id\_index

ON public.scooter USING btree

(scooter\_id ASC NULLS LAST)

TABLESPACE pg\_default;

-- Trigger: update\_state\_trigger

-- DROP TRIGGER IF EXISTS update\_state\_trigger ON public.scooter;

CREATE TRIGGER update\_state\_trigger

BEFORE UPDATE

ON public.scooter

FOR EACH ROW

EXECUTE FUNCTION public.update\_state\_on\_battery\_wear();

-- Table: public.model

-- DROP TABLE IF EXISTS public.model;

CREATE TABLE IF NOT EXISTS public.model

(

model\_id integer NOT NULL,

color character varying(30) COLLATE pg\_catalog."default" NOT NULL,

manufacturer\_country character varying(30) COLLATE pg\_catalog."default" NOT NULL,

manufacturer\_email character varying(30) COLLATE pg\_catalog."default",

warranty\_period integer,

CONSTRAINT model\_pkey PRIMARY KEY (model\_id),

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

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.model

OWNER to postgres;

-- Table: public.parking

-- DROP TABLE IF EXISTS public.parking;

CREATE TABLE IF NOT EXISTS public.parking

(

parking\_id integer NOT NULL,

warehouse\_id integer NOT NULL,

parking\_name character varying(50) COLLATE pg\_catalog."default" NOT NULL,

number\_places integer NOT NULL,

note character varying(200) COLLATE pg\_catalog."default",

CONSTRAINT parking\_pkey PRIMARY KEY (parking\_id),

CONSTRAINT parking\_warehouse\_key FOREIGN KEY (warehouse\_id)

REFERENCES public.warehouse (warehouse\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.parking

OWNER to postgres;

-- Table: public.warehouse

-- DROP TABLE IF EXISTS public.warehouse;

CREATE TABLE IF NOT EXISTS public.warehouse

(

warehouse\_id integer NOT NULL,

number\_personnel integer NOT NULL,

number\_places integer NOT NULL,

service\_cost money,

CONSTRAINT warehouse\_pkey PRIMARY KEY (warehouse\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.warehouse

OWNER to postgres;

-- Table: public.info\_employees

-- DROP TABLE IF EXISTS public.info\_employees;

CREATE TABLE IF NOT EXISTS public.info\_employees

(

employees\_id integer NOT NULL,

surname character varying(30) COLLATE pg\_catalog."default" NOT NULL,

name character varying(30) COLLATE pg\_catalog."default" NOT NULL,

patronymic character varying(30) COLLATE pg\_catalog."default",

birthdate date NOT NULL,

experience integer,

post character varying(200) COLLATE pg\_catalog."default" NOT NULL,

criminal\_record character varying(30) COLLATE pg\_catalog."default",

salary money NOT NULL,

CONSTRAINT employees\_pkey PRIMARY KEY (employees\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.info\_employees

OWNER to postgres;

-- Table: public.employees

-- DROP TABLE IF EXISTS public.employees;

CREATE TABLE IF NOT EXISTS public.employees

(

employees\_id integer NOT NULL,

warehouse\_id integer NOT NULL,

contract\_number integer,

wage\_rate integer,

CONSTRAINT employees\_prkey PRIMARY KEY (employees\_id, warehouse\_id),

CONSTRAINT employees\_warehouse\_key FOREIGN KEY (warehouse\_id)

REFERENCES public.warehouse (warehouse\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.employees

OWNER to postgres;

Код для заполнения базы данных:

--DELETE FROM scooter;

INSERT INTO scooter (scooter\_id, parking\_id, warehouse\_id, model\_id, latitude, longitude, charge\_lvl, battery\_wear, state)

VALUES

(100001, 200001, NULL, 400001, 55.9829, 37.2176, 70.0, 0.9, 'is used'),

(100002, 200002, NULL, 400002, 55.9944, 37.1947, 85.5, 0.85, 'waiting'),

(100003, NULL, 300003, 400003, 55.9835, 37.2179, 0.0, 0.8, 'discharged'),

(100004, 200004, NULL, 400004, 55.9774, 37.1719, 60.2, 0.75, 'is used'),

(100005, 200005, NULL, 400005, 55.9866, 37.1874, 77.3, 0.9, 'waiting'),

(100006, NULL, 300006, 400006, 55.9882, 37.2202, 0.0, 0.95, 'discharged'),

(100007, 200007, NULL, 400007, 55.9913, 37.2200, 55.9, 0.8, 'is used'),

(100008, 200008, NULL, 400008, 55.9907, 37.2092, 68.1, 0.85, 'waiting'),

(100009, NULL, 300009, 400009, 55.9962, 37.2108, 0.0, 0.9, 'discharged'),

(100010, 200010, NULL, 400010, 55.9879, 37.2056, 50.6, 0.75, 'is used'),

(100011, 200011, NULL, 400011, 55.9975, 37.2142, 63.8, 0.8, 'waiting'),

(100012, NULL, 300012, 400012, 55.9888, 37.2189, 0.0, 0.9, 'faulty');

--DELETE FROM model;

INSERT INTO model (model\_id, color, manufacturer\_country, manufacturer\_email, warranty\_period)

VALUES

(400001, 'Black', 'South Korea', 'office@e-twow.com', 4),

(400002, 'Purple', 'China', 'mail@xiaomi.com', 5),

(400003, 'Green', 'Russia', 'zakaz@e-samokat.com', 3),

(400004, 'Black', 'China', 'zakaz@e-samokat.com', 2),

(400005, 'Yellow', 'South Korea', 'office@e-twow.com', 4),

(400006, 'Yellow', 'China', 'info@ninebot.com', 3),

(400007, 'Black', 'Japan', 'info.business@eu.panasonic.com', 5),

(400008, 'Purple', 'China', 'info@segway.com', 3),

(400009, 'Green', 'South Korea', 'office@e-twow.com', 4),

(400010, 'Black', 'China', 'zakaz@e-samokat.com', 2),

(400011, 'Purple', 'Russia', 'zakaz@e-samokat.com', 5),

(400012, 'Black', 'Japan', 'info.business@eu.panasonic.com', 4),

(400013, 'Green', 'South Korea', 'office@e-twow.com', 3),

(400014, 'Purple', 'China', 'zakaz@e-samokat.com', 4),

(400015, 'Yellow', 'Japan', 'info.business@eu.panasonic.com', 5);

--DELETE FROM parking;

INSERT INTO parking (parking\_id, warehouse\_id, parking\_name, number\_places, note)

VALUES

(200001, 300001, 'Каштановая аллея, дом 1', 15, 'Рядом с домом 1 на улице Каштановая аллея'),

(200002, 300002, 'Площадь Колумба, дом 5', 20, 'Рядом с домом 5 на Площади Колумба'),

(200003, 300003, 'Улица Лётчика Полагушина, дом 10', 10, 'Рядом с домом 10 на улице Лётчика Полагушина'),

(200004, 300004, 'Крюковская площадь, дом 15', 25, 'Рядом с домом 15 на Крюковской площади'),

(200005, 300005, 'Кутузовское шоссе, дом 20', 30, 'Рядом с домом 20 на Кутузовском шоссе'),

(200006, 300006, 'Площадь Шокина, дом 25', 5, 'Рядом с домом 25 на Площади Шокина'),

(200007, 300007, 'Яблоневая аллея, дом 30', 18, 'Рядом с домом 30 на Яблоневой аллее'),

(200008, 300008, 'Площадь Юности, дом 35', 12, 'Рядом с домом 35 на Площади Юности'),

(200009, 300009, 'Улица Юности, дом 40', 8, 'Рядом с домом 40 на Улице Юности'),

(200010, 300010, 'Проспект Генерала Алексеева, дом 45', 22, 'Рядом с домом 45 на Проспекте Генерала Алексеева'),

(200011, 300011, 'Георгиевский проспект, дом 50', 14, 'Рядом с домом 50 на Георгиевском проспекте'),

(200012, 300012, 'Панфиловский проспект, дом 55', 7, 'Рядом с домом 55 на Панфиловском проспекте'),

(200013, 300013, 'Савёлкинский проезд, дом 60', 19, 'Рядом с домом 60 на Савёлкинском проезде'),

(200014, 300014, 'Солнечная аллея, дом 65', 11, 'Рядом с домом 65 на Солнечной аллее'),

(200015, 300015, 'Сосновая аллея, дом 70', 16, 'Рядом с домом 70 на Сосновой аллее');

--DELETE FROM warehouse;

INSERT INTO warehouse (warehouse\_id, number\_personnel, number\_places, service\_cost)

VALUES

(300001, 5, 50, 250000.0),

(300002, 7, 75, 300000.0),

(300003, 3, 30, 200000.0),

(300004, 6, 60, 275000.0),

(300005, 8, 80, 325000.0),

(300006, 4, 40, 225000.0),

(300007, 7, 70, 300000.0),

(300008, 5, 50, 250000.0),

(300009, 6, 60, 275000.0),

(300010, 8, 80, 325000.0),

(300011, 4, 40, 225000.0),

(300012, 5, 50, 250000.0),

(300013, 7, 70, 300000.0),

(300014, 3, 30, 200000.0),

(300015, 6, 60, 275000.0);

--DELETE FROM info\_employees;

INSERT INTO info\_employees (employees\_id, surname, name, patronymic, birthdate, experience, post, criminal\_record, salary)

VALUES

(1, 'Иванов', 'Иван', 'Иванович', '1990-05-15', 5, 'Менеджер по прокату', 'Нет', 50000.0),

(2, 'Петров', 'Петр', 'Петрович', '1985-08-22', 8, 'Инженер по техническому обслуживанию', 'Нет', 60000.0),

(3, 'Сидорова', 'Ольга', 'Ивановна', '1995-02-10', 2, 'Администратор проката', NULL, 40000.0),

(4, 'Козлов', 'Алексей', 'Алексеевич', '1950-11-18', 12, 'Директор', 'Нет', 80000.0),

(5, 'Смирнов', 'Дмитрий', 'Павлович', '1993-09-05', 4, 'Специалист по маркетингу', NULL, 55000.0),

(6, 'Кузнецова', 'Татьяна', 'Сергеевна', '1988-07-30', 7, 'Финансовый аналитик', NULL, 70000.0),

(7, 'Морозов', 'Артем', 'Артемович', '1998-04-12', 1, 'Специалист по клиентским отношениям', NULL, 35000.0),

(8, 'Лебедев', 'Александр', 'Владимирович', '1983-06-25', 10, 'HR-менеджер', 'Нет', 65000.0),

(9, 'Григорьев', 'Максим', 'Максимович', '1997-03-08', 3, 'IT-специалист', NULL, 60000.0),

(10, 'Васнецов', 'Евгений', 'Александрович', '1982-12-17', 11, 'Программист', NULL, 75000.0),

(11, 'Николаева', 'Светлана', 'Игоревна', '1991-10-20', 6, 'Дизайнер', NULL, 55000.0),

(12, 'Захаров', 'Антон', 'Сергеевич', '1987-09-03', 9, 'Менеджер по безопасности', NULL, 70000.0),

(13, 'Жуков', 'Денис', 'Петрович', '1996-06-14', 2, 'Тестировщик', NULL, 45000.0),

(14, 'Беляков', 'Игорь', 'Николаевич', '1986-08-28', 8, 'Инженер-проектировщик', NULL, 65000.0),

(15, 'Макаров', 'Владимир', 'Владимирович', '1994-03-07', 4, 'Специалист по обслуживанию электросамокатов', NULL, 50000.0);

INSERT INTO employees (employees\_id, warehouse\_id, contract\_number, wage\_rate)

VALUES

(1, 300001, 12345, 2.5),

(2, 300002, 67890, 1),

(3, 300003, 54321, 0.5),

(4, 300004, 98765, 1.75),

(5, 300005, 11111, 2),

(6, 300006, 22222, 1.5),

(7, 300007, 33333, 2),

(8, 300008, 44444, 1),

(9, 300009, 55555, 1.5),

(10, 300010, 66666, 2),

(11, 300011, 77777, 1.75),

(12, 300012, 88888, 0.5),

(13, 300013, 99999, 1.75),

(14, 300014, 11100, 2),

(15, 300015, 22200, 2.5);