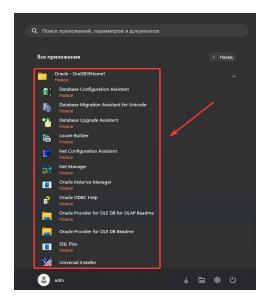
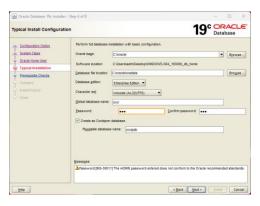
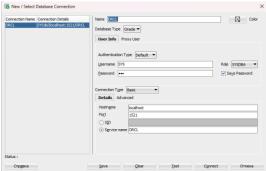
## Тестовое задание. <u>Часть 1.</u>

1. Установить Oracle Database Express Edition.



2. Разработать структуру и создать БД для сервиса сдачи автомобилей в аренду.





3. Основные сущности: клиенты, автомобили, заказы.

Учитывая, что используются только три сущности структура для сервиса сдачи автомобилей в аренду, будет выглядеть следующим образом:

### Таблица Customers (Клиенты):

- customer\_id (Уникальный идентификатор клиента);
- first name (Имя клиента);
- last name (Фамилия клиента);
- email (Адрес электронной почты клиента);
- phone (Номер телефона клиента):
- address (Адрес клиента).

### Таблица Cars (Автомобили):

- car id (Уникальный идентификатор автомобиля);
- make (Марка автомобиля);
- model (Модель автомобиля);
- year (Год выпуска автомобиля);
- registration (Регистрационный номер автомобиля);
- daily rate (Дневная ставка аренды);
- status (Статус автомобиля, например, "доступен", "арендован").

### Таблица Orders (Заказы):

- order\_id (Уникальный идентификатор заказа);
- customer id (Ссылка на клиента, сделавшего заказ);
- car\_id (Ссылка на арендованный автомобиль);
- start\_date (Дата начала аренды);
- end date (Дата окончания аренды);
- total\_cost (Итоговая стоимость аренды).

Следующим шагом создадим таблицу Customers, в соответствии с условием:

```
CREATE TABLE Customers (
customer_id NUMBER PRIMARY KEY,
first_name VARCHAR2(50),
last_name VARCHAR2(50),
email VARCHAR2(100),
phone VARCHAR2(20),
address VARCHAR2(255)
```

Проверим создание таблицы:

SELECT \* FROM Customers



Создадим аналогичным образом таблицу Cars, в соответствии с условием:

```
CREATE TABLE Cars (
car_id NUMBER PRIMARY KEY,
make VARCHAR2(50),
model VARCHAR2(50),
year NUMBER,
registration VARCHAR2(20) UNIQUE,
daily_rate NUMBER,
status VARCHAR2(20)
```

Добавлено примечание ([a1]): CREATE TABLE Customers: Эта часть SQL-команды начинает процесс создания новой таблицы с именем "Customers". Она определяет начало определения таблицы.

( customer\_id NUMBER PRIMARY KEY,: Здесь мы определяем первое поле "customer\_id". NUMBER - это тип данных, который предназначен для хранения числовых значений, в данном случае, идентификаторов клиентов. PRIMARY KEY указывает, что это поле является первичным ключом, что означает, что значения в этом столбце должны быть уникальными.

first\_name VARCHAR2(50),: Это определение второго поля "first\_name". VARCHAR2(50) означает, что это текстовое поле переменной длины (VARCHAR) с максимальной длиной в 50 символов. Это поле будет содержать имена клиентов.

last\_name VARCHAR2(50),: Это определение третьего поля "last\_name". То же, что и для "first\_name", это текстовое поле с максимальной длиной в 50 символов, предназначенное для фамилий клиентов.

email VARCHAR2(100),: Это четвертое поле "email". Также текстовое поле переменной длины с максимальной длиной в 100 символов, предназначенное для хранения адресов электронной почты клиентов.

phone VARCHAR2(20),: Это определение пятого поля "phone". Это текстовое поле переменной длины с

Добавлено примечание ([a2]): Команда SELECT \* FROM Customers - это SQL-команда, используемая для извлечения данных из таблицы "Customers" в базе данных.

SELECT: Это ключевое слово, которое указывает базе данных на необходимость выполнения операции выборки данных.

\*: Звездочка (\*) означает "все столбцы". Это обозначение говорит СУБД о том, что мы хотим выбрать все столбцы, содержащиеся в таблице "Customers".

FROM: Это ключевое слово, за которым следует имя таблицы, из которой мы хотим извлечь данные. В данном случае, имя таблицы – "Customers".

Добавлено примечание ([a3]): CREATE TABLE Cars: Эта часть SQL-команды начинает процесс создания новой таблицы с именем "Cars". Она определяет начало определения таблицы.

( car\_id NUMBER PRIMARY KEY,: Здесь мы определяем первое поле "car\_id". NUMBER - это тип данных, предназначенный для хранения числовых значений, в данном случае, идентификаторов автомобилей. PRIMARY KEY указывает, что это поле является первичным ключом, что гарантирует уникальность идентификаторов автомобилей в таблице.

make VARCHAR2(50),: Это определение второго поля "make". VARCHAR2(50) означает, что это текстовое поле переменной длины (VARCHAR) с максимальной длиной в 50 символов. Это поле будет содержать марку (бренд)

model VARCHAR2(50),: Это определение третьего поля "model". То же, что и для "make", это текстовое поле с максимальной длиной в 50 символов, предназначенное для модели автомобиля.

### Проверим создание таблицы:

### SELECT \* FROM Cars



Создадим таблицу Orders с внешними ключами, что позволит хранить данные о заказах и связывать их с соответствующими клиентами и автомобилями:

```
CREATE TABLE Orders (
order_id NUMBER PRIMARY KEY,
customer_id NUMBER,
car_id NUMBER,
start_date DATE,
end_date DATE,
total_cost NUMBER,
CONSTRAINT fk_customer
FOREIGN KEY (customer_id)
REFERENCES Customers(customer_id),
CONSTRAINT fk_car
FOREIGN KEY (car_id)
REFERENCES Cars(car_id)
);
```

# Проверим создание таблицы:

## SELECT \* FROM Orders



**Добавлено примечание ([a4]):** Тоже что и команда SELECT \* FROM Customers

Добавлено примечание ([a5]): CREATE TABLE Orders: Эта часть команды начинает процесс создания новой таблицы с именем "Orders".

order\_id NUMBER PRIMARY KEY: Здесь определены столбцы таблицы.

order\_id - это столбец с числовыми значениями, который будет использоваться в качестве первичного ключа таблицы. Первичный ключ уникален для каждой записи и обеспечивает уникальность каждого заказа в таблице. NUMBER - это тип данных столбца. В данном случае, это числовой тип данных.

customer\_id NUMBER и саr\_id NUMBER: Эти столбцы также представляют числовые значения и будут использоваться для хранения идентификаторов клиента и автомобиля, связанных с каждым заказом.

start\_date DATE и end\_date DATE: Эти столбцы хранят дату начала и окончания аренды автомобиля. Тип данных DATE используется для хранения дат.

total\_cost NUMBER: Этот столбец предназначен для хранения итоговой стоимости заказа и также имеет числовой тип данных.

CONSTRAINT fk\_customer FOREIGN KEY (customer\_id)
REFERENCES Customers(customer\_id): Эта часть команды
определяет внешний ключ для столбца customer\_id,
который связывает столбец customer\_id в таблице
"Orders" с первичным ключом customer\_id в таблице
"Customers". Это обеспечивает целостность данных и
позволяет связывать заказы с клиентами.

CONSTRAINT fk\_car FOREIGN KEY (car\_id) REFERENCES Cars(car\_id): Аналогично, это определяет внешний ключ для столбца car\_id, который связывает столбец car\_id в таблице "Orders" с первичным ключом car\_id в таблице "Cars". Это позволяет связывать заказы с автомобилями.

**Добавлено примечание ([а6]):** Тоже что и команда SELECT \* FROM Customers

#### 4. Реализовать:

а. Скрипты для наполнения БД тестовыми данными.

Для наполнения базы данных тестовыми данными в таблицу Customers внесем следующие данные:

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)
VALUES (1, 'Иван', 'Иванов', 'ivan.ivanov@email.com', '8-800-123-4567', 'Московская улица, 1, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (2, 'Екатерина', 'Петрова', 'ekaterina.petrova@email.com', '8-800-987-6543', 'Санкт-Петербургская улица, 2, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (3, 'Павел', 'Смирнов', 'pavel.smirnov@email.com', '8-800-555-5555', 'Красная площадь, 3, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (4, 'Анна', 'Козлова', 'anna.kozlova@email.com', '8-800-888-1234', 'Пресненская набережная, 4, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (5, 'Дмитрий', 'Медведев', 'dmitry.medvedev@email.com', '8-800-444-7777', 'Большая Дмитровка, 5, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (6, 'Светлана', 'Крылова', 'svetlana.krylova@email.com', '8-800-222-9999', 'Тверская улица, 6, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)
VALUES (7, 'Алексей', 'Соколов', 'alexei.sokolov@email.com', '8-800-111-3333', 'Казанская улица, 7,
Pocmoв-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (8, 'Мария', 'Волкова', 'maria.volkova@email.com', '8-800-999-2222', 'Невский проспект, 8, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (9, 'Cepzeü', 'Зайцев', 'sergei.zaitsev@email.com', '8-800-777-4444', 'Ленинский проспект, 9, Ростов-на-Дону');

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address) VALUES (10, 'Ольга', 'Лебедева', 'olga.lebedeva@email.com', '8-800-666-8888', 'Арбатская площадь, 10, Ростов-на-Дону');

3 З Павел Смирнов раvel.smirnov@email. 8-800-555-5555 Красная пловаль, 3, Ростов-на-Дону 4 4 Ания Колоза апа. kozlova@email.com 8-800-888-1234 Пресмеская имбережива, 4, Ростов-на-Дону 5 5 Дригури? Мелевсев смітту. дейсмеўчеса. 8-800-481-71234 Пресмеская имбережива, 4, Ростов-на-Дону 6 6 Светлама Крыкова зучеталь ктуlova@emai. 8-800-481-7173 Таврская улица, 6, Ростов-на-Дону 7 7. Алексей Соколов аleксі.soklov@emai. 8-800-111-3333 Казанская улица, 7, Ростов-на-Дону 8 8 Кмария Волкова ватія.volkova@email. 8-800-999-2222 Мескові проспект, 6, Ростов-на-Дону 9 9 9 Сергей Зайшев зегусі.заітаку@email. 8-800-777-4444 Левинский проспект, 9, Ростов-на-Дону 10 10 Ольта Лебелева olga.lebedeva@email 8-800-666-8888 Арбатская пловаль, 10, Ростов-на-Дону Намогу  4 4 Намогу	⊕ CUSTOMER_ID		LAST_NAME	() EMAIL	♦ PHONE		
3 3 Isaen Cosquos pavel.mirnov@email 8-800-555-5555 Rpacsas nnomans, 3, Pocros-sa-Ziony 4 4 4 Ausa Rossosa ana.kozlova@email.com 8-800-585-1314 Eprosecences ausdepezasa, 4, Pocros-sa-Ziony 5 5 Mpscrypti Mensezes 6 6 CBernasa Rymanosa vertana.krylova@em. 8-800-282-9999 Tespecas ymuta, 6, Pocros-sa-Ziony 7 7 Arenced Cosnone alessi.noko/demail 8-800-911-3333 Rassancas muta, 6, Pocros-sa-Ziony 8 8 Magus Bonxosa maria.volkova@email 8-800-911-3333 Rassancas ymuta, 7, Pocros-sa-Ziony 9 9 5 Cepred Safues sergei.naitzev@email 8-800-999-2222 Heschuid mpocnext, 8, Pocros-sa-Ziony 10 10 Onsra Jedenesa olga.lebedeva@email 8-800-666-8888 Apdarcas nnomans, 10, Pocros-sa-Ziony Whatty  **VINCO dostoners (sostoner.id, first_name, lest_name, emal, phone, address)VALUES (10, Onsra, Medaesa), Viga.bedeva@email.com, 8-800-777-4444 Messoncomi monusaba, 10, Pocros-sa-Ziony 17 NINCO dostoners (sostoner.id, first_name, lest_name, emal, phone, address)VALUES (10, Onsra, Medaesa), Viga.bedeva@email.com, 8-800-777-4444 Messoncomi monusaba, 10, Pocros-sa-Jiony 17 NINCO dostoners (sostoner.id, first_name, lest_name, emal, phone, address)VALUES (10, Onsra, Medaesa), Viga.bedeva@email.com, 8-800-777-4444 Messoncomi monusaba, 10, Pocros-sa-Jiony 17 NINCO dostoners (sostoner.id, first_name, lest_name, emal, phone, address)VALUES (0, Corred), Tabalesa, viga.bedeva@email.com, 8-800-777-4444 Messoncomi monusaba, 10, Pocros-sa-Jiony 17 NINCO dostoners (sostoner.id, first_name, lest_name, emal, phone, address)VALUES (0, Corred), Tabalesa, viga.bedeva@email.com, 8-800-977-9444, Messoncomi monusaba, 10, Pocros-sa-Jiony 17 NINCO dostoners (sostoner.id, first_name, lest_name, emal, phone, address)VALUES (0, Nomaesa, Messonsabenal.com, 8-800-977-9444, Messoncomi monusaba, 10, Pocros-sa-Jiony 17 NINCO dostoners (sostoner.id, first_name, lest_name, emal, phone, address)VALUES (0, Nomaesa, Nomaesa, Messa-Nova@email.com), 8-800-977-74447, Messoncomi monusaba, 10, Pocros-sa-Jion, ORCL 17 NINCO dostoners (sostoner.id, first_name, lest_name, emal,	1	1 Иван	Иванов	ivan.ivanov@email.com	8-800-123-4567	Московская улица, 1, Ростов-на-Дон	У
4 4 Alassa Holdone Anna kollova@email.com 8-800-888-1234 Пресмежская мабережная, 4, Ростов-на-Дону 5 5 (Вретурий Метелев dmitty medvedev@email.com. 8-800-484-7777 Воливая Драгурова, 5, Ростов-на-Дону 6 6 (Сведная Кранова wvetlana.tytynodemu. 8-800-484-7777 Воливая Драгурова, 5, Ростов-на-Дону 7 7 Алексей Соколов alsesi.sokolov@email 8-800-213-333 Казакская улица, 7, Ростов-на-Дону 8 8 Маркия Воликова магія voltov@email 8-800-313-333 Казакская улица, 7, Ростов-на-Дону 9 9 9 9 Сергей Safues sergei.naitaev@email 8-800-777-444 Деникский проспект, 8, Ростов-на-Дону 10 10 Полъта Дебедева olga.lebedeva@email 8-800-777-444 Деникский проспект, 9, Ростов-на-Дону 10 Спотов дебедера онд 10 Спотов дебеде	2	2 Екатерина	Петрова	ekaterina.petrova@e	8-800-987-6543	Санкт-Петербургская улица, 2, Ростов-на-Дон	
5 S JESTPURÍ NERBERES dmitry.medvedevêema 8-800-444-7777 BORDBAR JESTPORRA, S, POCTOS-HAS-JONY 6 GCB-STARKA NGAMBAR SYSTEMAN S	3	3 Павел	Симрнов	pavel.smirnov@email	8-800-555-5555	Красная площадь, 3, Ростов-на-Дону	
6 6 CRETABAN RAMONDA AVECTABLE TO THE CONTROL OF T	4	4 Анна	Козлова	anna.kozlova@email.com	8-800-888-1234	Пресненская набережная, 4, Ростов-на-Дону	
7 America Connect Connect Banaria voltovalemaii 8-800-111-3333 Kasamiraa ymmua, 7, Boctos-ma-Jony 8 Banaria Bonnosa maria voltovalemaii 8-800-599-2222 Beacond mpocnect, 8, Boctos-ma-Jony 10 10 mbra Jefezesa olga-lebedevalemaii 8-800-666-8888 Ap&atcas mnomana, 10, Poctos-ma-Jony 110 mbra Jefezesa olga-lebedevalemaii 8-800-666-8888 Ap&atcas mnomana, 10, Poctos-ma-Jony 118tory    Intro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Onera', Tefezesa', diga-lebedevalemai.com', 8-800-777-4444, Thesencosi mpocnect, 9, Poctos-ma-Jony) 11 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Onera', Tefezesa', diga-lebedevalemai.com', 8-800-666-8888 Ap&atcas mnomana, 10, Poctos-ma-Jony) 11 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Onera', Tefezesa', Septe. zattev-demal.com', 8-800-777-4444', Thesencosi mpocnect, 9, Poctos-ma-Jony) 11 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Consea', Tema-Jony) 12 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Consea', Tema-Jony) 13 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Consea', Tema-Jony) 14 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Consea', Tema-Jony) 15 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Consea', Tema-Jony) 15 Thro Customers (unstomer Jd, first_mame, lest_mame, emal, phone, address)/ALUES (10, Tema-Jd, Nemocal, "mame, Jd, Nemocal," Nemocal, "Nemocal," Nemocal, "Nemocal," Nemocal, "Nemocal," Nemocal, "Nemocal, "Nemocal," Nemocal, "Nemocal, "Nemocal," Nemocal, "Nemocal, "Nemocal," Nemocal, "Nemocal," Nemocal, "Nemocal, "Nemocal," Nemocal, "Nemocal, "Nemoca	5	5 Дмитрий	Медведев	dmitry.medvedev@ema	8-800-444-7777	Большая Диитровка, 5, Ростов-на-Дону	
8 SMapsus Bonnosa maria.voltova@emali8-800-999-2222 Heacksdi mpocnext, 8, Poctos-Ha-Zoxy 9 9 Scpred Safues sergei.zaitzev@emali8-800-777-4444 Remuncudi mpocnext, 9, Poctos-Ha-Zoxy 10 10 Ombra Jedenes olgs.lebedeva@emali8-800-666-8888 Ap@arcass.momaama, 10, Poctos-Ha-Zoxy Helstory  **Monocomer (Loutoner (Lo	6	6 Светлана	Крылова	svetlana.krylova@em	8-800-222-9999	Тверская улица, 6, Ростов-на-Дону	
9 9 9 Cepreii Safiase sergei.raitsev@emai 8-80-9777-4444   Damacomii ppocnear, 9, Bocros-sta-Tjony 10 10 Emar   The Casternes (Lustomer (Lustom	7	7 Алексей	Соколов	alexei.sokolov@emai	8-800-111-3333	Казанская улица, 7, Ростов-на-Дону	
10 Otto pra Decrease olganical control of the Language and Connect Con	8	8 Мария	Волкова	maria.volkova@email	8-800-999-2222	Невский проспект, 8, Ростов-на-Дону	
Introduction Customers (customer jd. first_name, last_name, emal, phone, address)/ALUES (10, 'Onera', 'Tefegaess', 'dga lebeder a Benal.com', '8-900-666-8888', 'Ap6arroas nnousata, 10, Poctroe +a J	9	9 Сергей	Зайцев	sergei.zaitsev@emai	8-800-777-4444	Ленинский проспект, 9, Ростов-на-Д	ону
Connect T DITO Customers (customer_id, first_name, lost_name, email, phone, address)VALLES (10, Yoner's', Trefeageas', 'diga lebederva@email.com', '8-800-666-8888', 'Ap6arocas nnowaas, 10, Portros +a-3_L COR. T BITO Customers (customer_id, first_name, lost_name, email, phone, address)VALLES (9, Yepres', '3-shaze'), bergis_rate-v@email.com', '8-800-777-4444', 'Newnoosin procnectr, 3, Portros +a-3_Lenvy'); GRC. T BITO Customers (customer_id, first_name, lost_name, email, phone, address)VALLES (8, Napasi, 'Bonocas', instead solvious@email.com', '8-800-2792-272, 'Heanoosin procnectr, 3, Portros +a-3_Lenvy'); GRC. T BITO Customers (customer_id, first_name, lost_name, email, phone, address)VALLES (6, Ceername', Yearnosa', Yearnosa	10	10 Ольга	Лебедева	olga.lebedeva@email	8-800-666-8888	Арбатская площадь, 10, Ростов-на-Д	ону
IT NTO Castomers (natomer J. fi. first_neme, leat_neme, emal, phone, address)/ALLES (10, Tours /, Tefecareas, joigs labedeva@menial com; 3-80-07-6488; jedeforcas nnowates, 10, Portone+sp.— ORC. IT NTO Castomers (natomer J. first_neme, leat_neme, emal, phone, address)/ALLES (s. Cycer's), "abuset, 'yeops zatiseve-leand com; 3-80-07-7444f, "hereovisor procerts, p. Portone+sp.—); ORC. IT NTO Castomers (natomer J. first_neme, leat_neme, emal, phone, address)/ALLES (s. Natoris, 'Boncoas', main voltone@menia.com', 3-80-09-2224f, Nessocia yrous, p. Fortone+sp.—); ORC. IT NTO Castomers (natomer J. first_neme, leat_neme, emal, phone, address)/ALLES (s. Cierrana', Yeoproas', 'leven splotley'); Associa yrous, p. Fortone+sp.—(portone-sp.); ORC. IT NTO Castomers (natomer J. first_neme, leat_neme, emal, phone, address)/ALLES (s. Cierrana', Yeoproas', 'leven splotley'); Associa yrous, p. Fortone+sp.—(portone-sp.); ORC. IT NTO Castomers (natomer J. first_neme, leat_neme, emal, phone, address)/ALLES (s. Terrana', Yeoproas', 'vent yeolow'); Associa yrous, p. Fortone+sp., oRc. IT NTO Castomers (natomer J. first_neme, leat_neme, emal, phone, address)/ALLES (s. Nemer), 'Nemersea', 'vent yeolow'); Associa yeolow', and	<b>3</b>						Connection
TINTO Customers (austomer j.d., first_name, leat_name, emal, phone, address)/ALLES (9, Yoprox", Sakueri, 'seeps attaive-demalacon', 8-800-977-4444', 'Thesencou'in process, 7-9 Poctore-s-rjow/f); GRC ITRIO Customers (automer j.d., first_name, leat_name, emal, phone, address)/ALLES (8, Massi, Sanosea), misks voldows@mestacon', 8-800-977-4444', Thesencou'in process, 7-9 Poctore-s-rjow/f); GRC ITRIO Customers (automer j.d., first_name, leat_name, emal, phone, address)/ALLES (7, Poctore', Yournore', Selves.sokiolv@emal.com', 8-900-111-3333', Kasonocoa yraus, 7, Poctore-s-rjow/f); GRC ITRIO Customers (automer j.d. first_name, leat_name, emal, phone, address)/ALLES (5, Piserpox', Messencee', 'mistr_name/develmel.com', 8-900-222-9999', Treposa yraus, 9, Poctore-s-rjow GRC ITRIO Customers (automer j.d., first_name, leat_name, emal, phone, address)/ALLES (5, Piserpox', Messencee', 'mistr_name/develmel.com', 8-900-885-1224', Thecencoas nadepsexae, 4, Poctore-s-rjow GRC ITRIO Customers (automer j.d. first_name, leat_name, emal, phone, address)/ALLES (4, Piser), Konzoea', 'arna kadova@emal.com', 8-900-885-1224', Thecencoas nadepsexae, 4, Poctore-s-rjow GRC ITRIO Customers (automer j.d. first_name, leat_name, emal, phone, address)/ALLES (3, Theor, Yourney', "pised-simovo@emal.com', 8-900-885-1224', Thecencoas nadepsexae, 4, Poctore-s-rjow GRC ITRIO Customers (automer j.d. first_name, leat_name, emal, phone, address)/ALLES (3, Theor, Yourney', "pised-simovo@emal.com', 8-900-885-1224', Theorencoas nadepsexae, 4, Poctore-s-rjow GRC ITRIO Customers (automer j.d. first_name, leat_name, emal, phone, address)/ALLES (3, Theor, Yourney', "pised-simovo@emal.com', 8-900-881-1224', Theorencoas nadepsexae, 4, Poctore-s-rjow GRC ITRIO Customers (automer j.d. first_name, leat_name, emal, phone, address)/ALLES (3, Theor, Yourney', "pised-simovo@emal.com', 8-900-881-1224', Theorencoas nadepsexae, 4, Poctore-s-rjow GRC ITRIO Customers (automer j.d. first_name, leat_name, emal, phone, address)/ALLES (3, Theor, Yourney', pi		RT INTO Customers (customer id., first name, last name, email, phone, address)VALUES (10, 'Onera', 'DeGenesa', 'oloa,lebedeva@email.com', '8-800-666-8888', 'Apparoxas плошадь, 10, Ростов на-Д					
TRITO Catamers (automer (d, first_jame, last_jame, emal, phone, addres)/ALLES (B, Ngaris, 'Boncoas', mais-audova@emal.com', '8-80-99-2227, 'Hesooi mpomers', 8-Portone-s-(Borry)', GCL TRITO Catamers (automer (d, first_jame, last_jame, emal, phone, addres)/ALLES (6, Nearyai', Nearyai', 'elses-alaciovil' espellacion', '8-800-1113-333', 'Assancioas yraus, 9, Portone-s-(Borry)', GCL TRITO Catamers (automer (d, first_jame, last_jame, emal, phone, addres)/ALLES (6, Nearyai', Nea	T INTO Customers (custome	r id, first name, last name, email	. phone. address)VALUES	(10. 'Ольга', 'Лебедева', 'olga,lebede			ORCL
I RITO Customers (dustomers (dustomers (dustomers (dustomers (dustomers (dustomers (dustomers (dustomers) (http://dustomers.)							
I INTO Customers (austomer Jd, first_name, lest_name, emal, phone, addres)/ALLES (5, ]javerpoi, Measeaseri, *dmin; methodes/@emal.com; 78:00-444-7777, "Sonsuas jiumposra, 5, Portose-n. ORC. I INTO Customers (austomer Jd, first_name, last_name, emal, phone, addres)/ALLES (4, *Aeva', Xousses', arma\u00edadors) @emal.com; 78:00-688-1224, Tipocentosasas@epossas, 4, Portose-ng-Jo ORC. I INTO Customers (austomer Jd, first_name, last_name, addres)/ALLES (3, Taseri, Youpper), 'pavel_aminrov(@emal.com', 8:400-585:555, Youcses nroussas, 5, Portose-ng-Jon-Y). ORC.	INTO Customers (custome	er_id, first_name, last_name, email	, phone, address)VALUES	(9, 'Сергей', 'Зайцев', 'sergei.zaitsev(	@email.com', '8-800-777-	4444', 'Ленинский проспект, 9, Ростов-на-Дону');	ORCL
EINTO Customers (customer jd., first_name, last_name, emal, phone, address)VALUES (4, 'Aнна', Kosnosa', 'anna kozlova@email.com', '8-800-888-1224', Tipecненская набережная, 4, Ростов-на-До ORCL EINTO Customers (customer_id, first_name, last_name, email, phone, address)VALUES (3, 'Tasen', 'Owiphod', 'pavel.smirnov@email.com', '8-800-585-5555', Kpachas площадь, 3, Ростов-на-Дону); ORCL	FINTO Customers (custome FINTO Customers (custome	er_id, first_name, last_name, email er_id, first_name, last_name, email	, phone, address)VALUES , phone, address)VALUES	(9, 'Сергей', 'Зайцев', 'sergei.zaitsev( (8, 'Мария', 'Волкова', 'maria.volkovai	@email.com', '8-800-777- @email.com', '8-800-999	4444', 'Ленинский проспект, 9, Ростов на-Дону'); 2222', 'Невский проспект, 8, Ростов на-Дону');	ORCL ORCL
TINTO Customers (customer_id, first_name, last_name, email, phone, address)VALUES (3, Tlasen', 'Смирнов', 'рavel.smirnov@email.com', '8-800-555-5555', 'Красная площадь, 3, Ростов на-дону'); ORCL	T INTO Customers (custome T INTO Customers (custome T INTO Customers (custome T INTO Customers (custome T INTO Customers (custome	er_id, first_name, last_name, email er_id, first_name, last_name, email er_id, first_name, last_name, email er_id, first_name, last_name, email	, phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES	(9, 'Сергей', 'Зайцев', 'sergei.zaitsev( (8, 'Мария', 'Волкова', 'maria.volkovai (7, 'Алексей', 'Соколов', 'alexei.sokok (6, 'Светлана', 'Крылова', 'svetlana.k	Bemail.com', '8-800-777- Bemail.com', '8-800-999- ov@email.com', '8-800-1 rylova@email.com', '8-80	4444', 'Ленинский проспект, 9, Ростов-на-Дону'); -2222', 'Невский проспект, 8, Ростов-на-Дону'); 11-3333', 'Казанская улица, 7, Ростов-на-Дону'); 00-222-9999', 'Тверская улица, 6, Ростов-на-Дон	ORCL ORCL ORCL
	RT INTO Customers (custome RT INTO Customers (custome RT INTO Customers (custome RT INTO Customers (custome	er_id, first_name, last_name, email er_id, first_name, last_name, email er_id, first_name, last_name, email er_id, first_name, last_name, email	, phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES	(9, 'Сергей', 'Зайцев', 'sergei.zaitsev( (8, 'Мария', 'Волкова', 'maria.volkovai (7, 'Алексей', 'Соколов', 'alexei.sokok (6, 'Светлана', 'Крылова', 'svetlana.k	Bemail.com', '8-800-777- Bemail.com', '8-800-999- ov@email.com', '8-800-1 rylova@email.com', '8-80	4444', 'Ленинский проспект, 9, Ростов-на-Дону'); -2222', 'Невский проспект, 8, Ростов-на-Дону'); 11-3333', 'Казанская улица, 7, Ростов-на-Дону'); 00-222-9999', 'Тверская улица, 6, Ростов-на-Дон	OR O
INTO Customers (quetomers id first name last name anali phone address)VALLES (2 "Suzzanaus" "Dazonaus" (alcaterina petrous Romal com" '9,900,097,4540' ("caust Dazon Surgan 2") (000)	INTO Customers (custome INTO Customers (custome INTO Customers (custome INTO Customers (custome INTO Customers (custome	er id, first name, last name, email er id, first name, last name, email	, phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES	(9, 'Сергей', 'Зайцев', 'sergei.zaitsev( (8, 'Мария', 'Волкова', 'maria.volkovai (7, 'Алексей', 'Соколов', 'alexei.sokok (6, 'Светлана', 'Крылова', 'svetlana.k (5, 'Диитрий', 'Медведев', 'dmitry.me	Bemail.com', '8-800-777- Bemail.com', '8-800-999- ov@email.com', '8-800-1 trylova@email.com', '8-80 dvedev@email.com', '8-80	4444, 'Леняновий проспект, 9, Ростов на-Дону'); 2222, 'Невокий проспект, 8, Ростов на-Дону'); 11-3333', 'Казанокая уямца, 7, Ростов на-Дону'); 30-222-9999, 'Тверская уямца, 6, Ростов на-Дон' 800-444-7777', 'Большая Динтровка, 5, Ростов н	ORCL ORCL ORCL ORCL ORCL
N INFO Customer's (customer id, first name, last_name, enan, prome, adures) values (z. Eval epiwar, nier poer, excelenta, perova genan, com, s. excelenta, esc.), description, enan, prome, adures) values (z. Eval epiwar, nier poer, excelenta, perova genan, com, s. excelenta, esc.), description, com, s. excelenta, esc., description, com, s. excelenta, esc., description, enan, esc., description, esc., desc., description, esc., description, esc., description, esc., des	RT INTO Customers (custome RT INTO Customers (custome	er jd, first_name, last_name, email er jd, first_name, last_name, email	, phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES , phone, address)VALUES	(9, 'Сергей', 'Зайцев', 'sergel.zaitsev( (8, 'Мария', 'Волкова', 'maria.volkova' (7, 'Алексей', 'Соколого', 'alexei.sokol (6, 'Светлана', 'Крылова', 'svetlana.k (5, 'Двитрий', 'Медведев', 'dmitry.me (4, 'Анна', 'Козлова', 'anna.kozlova@«	@email.com', '8-800-777- @email.com', '8-800-999- ov@email.com', '8-800-1: trylova@email.com', '8-80- edvedev@email.com', '8-80-888-1: email.com', '8-800-888-1:	4444°, Леминский проспект, 9, Ростов на-Дону); 2222°, Уневский проспект, 8, Ростов на-Дону); 11-3333°, Казанская уляца, 7, Ростов на-Дону); 00-222-9999°, Тверская уляца, 6, Ростов на-Дон- 00-444-7777°, Тольшая Динтровка, 5, Ростов на- 234°, Преспенская набережива, 4, Ростов на-Дон-	ORCL ORCL ORCL ORCL ORCL ORCL

Добавлено примечание ([a7]): Для наполнения базы данных тестовыми данными с использованием SQL, мы будем использовать команды (NSERT INTO.

Синтаксис команды INSERT INTO

INSERT INTO table\_name (column1, column2, column3, ...)

Где:

INSERT INTO: Это ключевое слово, которое указывает на начало операции вставки данных в таблицу. table\_name: Это имя таблицы, в которую мы хотим вставить данные.

вставить данные. (column1, column2, column3, ...): Это перечисление столбцов, в которые мы хотим вставить данные. Мы можем указать только те столбцы, которые нужно заполнить, остальные столбцы могут иметь значения по умолчанию или быть NULL, если не указаны. VALUES (value1, value2, value3, ...): Это перечисление

значений, которые мы хотим вставить в соответствующие столбцы. Порядок значений должен соответствовать порядку столбцов. Для наполнения базы данных тестовыми данными в таблицу Cars внесем следующие ланные:

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (1, 'Toyota', 'Camry', 2020, 'A111AA123', 50.00, 'доступен');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (2, 'Honda', 'Civic', 2019, 'A222BB123', 45.00, 'docmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (3, 'Ford', 'Escape', 2021, 'E777KK123', 55.00, 'доступен');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (4, 'Chevrolet', 'Malibu', 2018, 'M333OO456', 40.00, 'docmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (5, 'Volkswagen', 'Passat', 2022, 'T555EE123', 60.00, 'docmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (6, 'Nissan', 'Altima', 2019, 'A999AA123', 45.00, 'доступен');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (7, 'Kia', 'Sorento', 2021, 'P777VV123', 55.00, 'oocmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (8, 'Hyundai', 'Tucson', 2020, '012300123', 52.00, '00cmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (9, 'Mercedes-Benz', 'E-Class', 2022, 'V555VV123', 80.00, 'oocmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (10, 'Audi', 'A4', 2023, '099900123', 75.00, '0ocmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (11, 'BMW', 'X5', 2021, 'P888VV123', 70.00, 'oocmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (12, 'Lexus', 'RX', 2020, 'A777AA123', 65.00, 'oocmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (13, 'Volvo', 'XC90', 2022, 'M111MM123', 70.00, 'доступен');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (14, 'Subaru', 'Outback', 2019, 'H444HH123', 55.00, 'доступен');

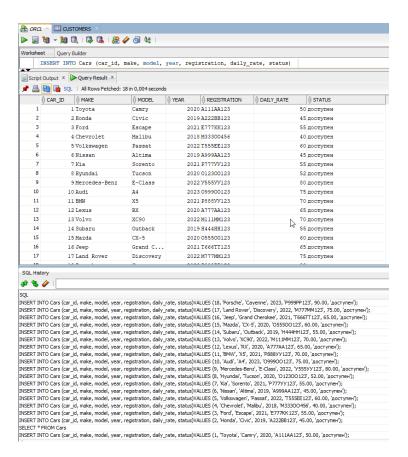
INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (15, 'Mazda', 'CX-5', 2020, 'O555OO123', 60.00, 'docmyneh');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (16, 'Jeep', 'Grand Cherokee', 2021, 'T666TT123', 65.00, 'docmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)
VALUES (17, 'Land Rover', 'Discovery', 2022, 'M777MM123', 75.00, 'docmynen');

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (18, 'Porsche', 'Cayenne', 2023, 'P999PP123', 90.00, 'docmynen');

**Добавлено примечание ([a8]):** Тоже что и команда INSERT INTO для Customers

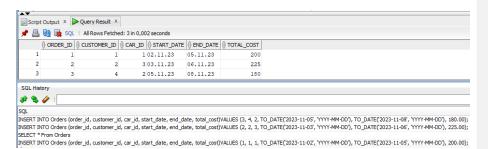


Для наполнения базы данных тестовыми данными в таблицу Orders внесем следующие ланные:

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (1, 1, 1, TO\_DATE('2023-11-02', 'YYYY-MM-DD'), TO\_DATE('2023-11-05', 'YYYY-MM-DD'), 200.00):

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (2, 2, 3, TO\_DATE('2023-11-03', 'YYYY-MM-DD'), TO\_DATE('2023-11-06', 'YYYY-MM-DD'), 225.00);

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)
VALUES (3, 4, 2, TO\_DATE('2023-11-05', 'YYYY-MM-DD'), TO\_DATE('2023-11-08', 'YYYY-MM-DD'),
180.00);



Добавлено примечание ([a9]): INSERT INTO: Ключевое слово INSERT INTO указывает на то, что вы собираетесь вставить новую строку данных в таблицу

Table Name: Это имя таблицы, в которую мы хотим вставить новую строку данных.

Column Names: Если мы явно указываем столбцы, в которые мы собираемся вставить данные, то после имени таблицы следует перечислить имена столбцов в скобках.

VALUES: Ключевое слово VALUES указывает на начало списка значений, которые мы собираемся вставить в

Data Values: Значения, которые мы вставляем, должны соответствовать типам данных столбцов, в которые они вставляются. Если столбцы имеют числовые значения, то числовые значения должны быть представлены без кавычек, а если столбцы имеют строковые значения, то строковые значения должны быть заключены в одинарные кавычки (" ").

Дата и время: Если столбцы таблицы имеют тип данных даты и времени, то даты должны быть представлены в формате, который соответствует формату даты и времени в нашей базе данных. Используется функция TO\_DATE для преобразования строкового значения в тип данных даты

b. CRUD операции для всех сущностей.

### CRUD операции для сущности Customers:

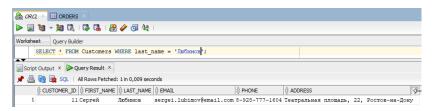
Create (Создание):

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)
VALUES (11, 'Сергей', 'Любимов', 'sergei.lubimov@email.com', '8-928-777-1604', 'Театральная площадь,
22, Ростов-на-Дону');



## Read (Чтение):

SELECT \* FROM Customers WHERE last name = 'Любимов';



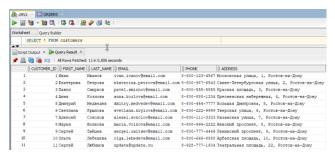
## Update (Обновление):

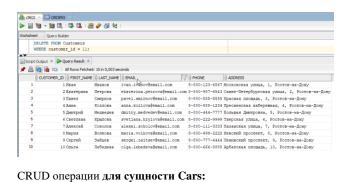
UPDATE Customers SET email = 'update@update.ru' WHERE customer\_id = 11;



## Delete (Удаление):

DELETE FROM Customers WHERE customer\_id = 11;

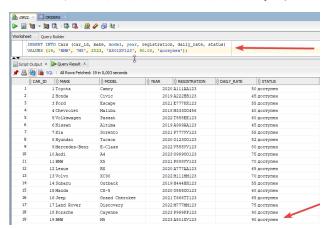




## CRUD операции для сущности Cars:

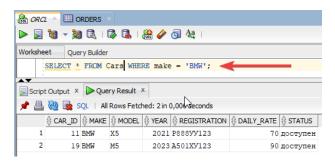
Create (Создание):

INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status) VALUES (19, 'BMW', 'M5', 2023, 'A501XV123', 90.00, 'docmyneh');



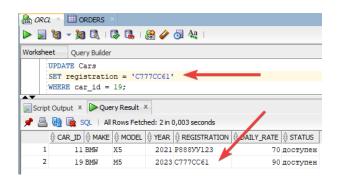
# Read (Чтение):

SELECT \* FROM Customers WHERE make = 'BMW';



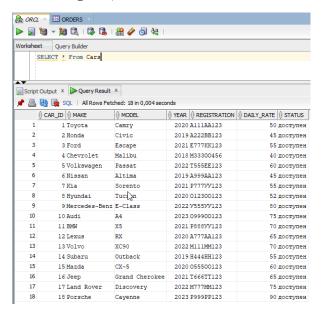
# Update (Обновление):

UPDATE Cars SET registration = C777CC61 WHERE  $car_id = 19$ ;



## Delete (Удаление):

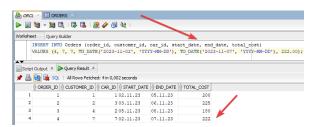
DELETE FROM Cars WHERE car\_id = 19;



# CRUD операции для сущности Orders:

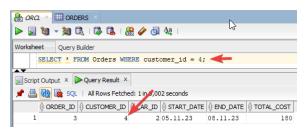
Create (Создание):

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (4, 7, 7, TO\_DATE('2023-11-02', 'YYYY-MM-DD'), TO\_DATE('2023-11-07', 'YYYY-MM-DD'), 222.00);



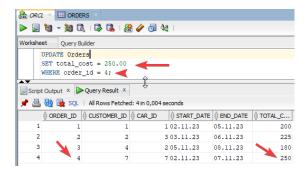
## Read (Чтение):

SELECT \* FROM Orders WHERE customer\_id = 4;



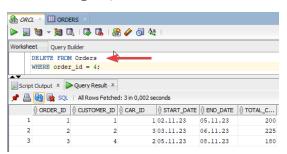
## Update (Обновление):

UPDATE Orders SET total\_cost = 250.00 WHERE order\_id = 4;



### Delete (Удаление):

DELETE FROM Orders WHERE order\_id = 4;

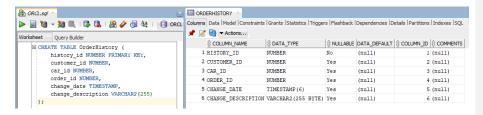


с. сохранение истории изменения заказов.

Для сохранения истории изменений заказов в базе данных нам потребуется создать таблицу для хранения истории изменений, создать последовательности и триггер, который будет записывать изменения в эту таблицу:

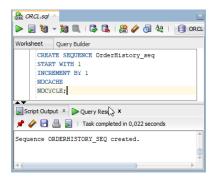
### Создание таблицы:

```
CREATE TABLE OrderHistory (
history_id NUMBER PRIMARY KEY,
customer_id NUMBER,
car_id NUMBER,
order_id NUMBER,
change_date TIMESTAMP,
change_description VARCHAR2(255)
```



### Создание последовательности:

CREATE SEQUENCE OrderHistory\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;



'Order updated'):

Создание триггера для отслеживания изменений:

```
CREATE OR REPLACE TRIGGER ORDERSHISTORYTRIGGER
AFTER INSERT OR UPDATE OR DELETE ON Orders
FOR EACH ROW
BEGIN
IF INSERTING THEN
INSERT INTO OrderHistory (history_id, order_id, customer_id, car_id, change_date, change_description)
VALUES (OrderHistory_seq.NEXTVAL, :new.order_id, :new.customer_id, :new.car_id, SYSTIMESTAMP,
'Order created');
ELSIF UPDATING THEN
INSERT INTO OrderHistory (history_id, order_id, customer_id, car_id, change_date, change_description)
VALUES (OrderHistory_seq.NEXTVAL, :new.order_id, :new.customer_id, :new.car_id, SYSTIMESTAMP,
```

**Добавлено примечание ([a10]):** Таблица "OrderHistory," будет хранить историю изменений заказов.

history\_id (уникальный идентификатор записи в истории, первичный ключ). customer\_id (ссылка на клиента, совершившего заказ). car id (ссылка на автомобиль в заказе).

order\_id (ссылка на исходный заказ). change\_date (дата и время изменения). change\_description (описание изменения заказа, например, "создание заказа," "обновление заказа").

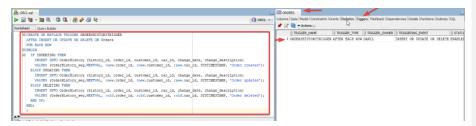
Добавлено примечание ([a11]): Последовательность (sequence), которая будет использоваться для генерации уникальных значений для столбца history\_id в таблице "OrderHistory."

#### ELSIF DELETING THEN

INSERT INTO OrderHistory (history\_id, order\_id, customer\_id, car\_id, change\_date, change\_description) VALUES (OrderHistory\_seq.NEXTVAL, :old.order\_id, :old.customer\_id, :old.car\_id, SYSTIMESTAMP, 'Order deleted');

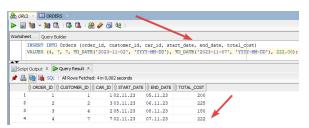
END IF;

END;



Чтобы проверить сохранение истории заказов добавим информацию об аренде:

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (4, 7, 7, TO\_DATE('2023-11-02', 'YYYY-MM-DD'), TO\_DATE('2023-11-07', 'YYYY-MM-DD'), 222.00);



Убедимся, что изменения, сделанные в таблице Orders отображены в таблице OrderHistory:



Добавлено примечание ([a12]): CREATE OR REPLACE TRIGGER ORDERSHISTORYTRIGGER - это начало команды создания триггера. Она начинается с ключевого слова CREATE, что указывает на создание нового триггера. Фраза OR REPLACE указывает, что если триггер с таким именем уже существует, он будет перезаписан новым триггером с тем же именем. ORDERSHISTORYTRIGGER - это имя нашего триггера.

AFTER INSERT OR UPDATE OR DELETE ON Orders - это часть, которая определяет, когда триггер будет срабатывать. В данном случае, триггер срабатывает после операций вставки (INSERT), обновления (UPDATE) и удаления (DELETE) в таблице Orders. Таким образом, триггер будет реагировать на любые изменения в нашей таблице.

FOR EACH ROW - это часть, которая указывает, что триггер будет выполняться для каждой строки, затронутой операцией. Внутри блока триггера можно обращаться к данным строки, которую затронула операция.

BEGIN - начало блока триггера. Все операции триггера размещаются между BEGIN и END.

БЛОК IF INSERTING THEN ... ELSIF UPDATING THEN ... ELSIF DELETING THEN ... END IF; - это условные операторы, которые определяют, какое действие было выполнено (вставка, обновление или удаление). В зависимости от действия выполняется соответствующая операция записи в таблицу OrderHistory.

В каждом блоке IF используются операторы INSERT INTO, чтобы добавить запись в таблицу OrderHistory. Каждая запись содержит следующие значения:

history\_id - значение, которое генерируется из последовательности OrderHistory\_seq. customer\_id, саr\_id, order\_id - значения, полученные из данных строки, которую затронула операция. change\_date - текущая дата и время, полученные с помощью функции SYSTIMESTAMP. change\_description - описание изменения (например, "Order created," "Order updated," "Order deleted"). / - это окончание команды.

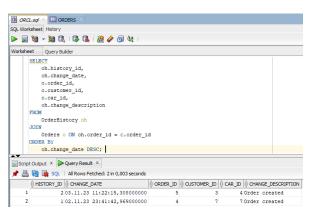
d. отчет об истории изменения заказов.

## Выполним запрос:

```
SELECT
oh.history_id,
oh.change_date,
o.order_id,
o.customer_id,
o.car_id,
oh.change_description
FROM
OrderHistory oh
JOIN
Orders o ON oh.order_id = o.order_id
ORDER BY
oh.change_date DESC;
```

Чтобы проверить работу отчета об истории изменения заказов добавим информацию об аренде:

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (5, 3, 4, TO\_DATE('2023-11-03', 'YYYY-MM-DD'), TO\_DATE('2023-11-08', 'YYYY-MM-DD'), 100.00);



Добавлено примечание ([a13]): SQL-запрос выбирает столбцы из таблицы "OrderHistory" и "Orders," включая history\_id, change\_date, order\_id, customer\_id, car\_id и change\_description.

Использует оператор JOIN, чтобы связать таблицы "OrderHistory" и "Orders" по полю order\_id, чтобы получить информацию о заказе, связанном с каждой записью истории.

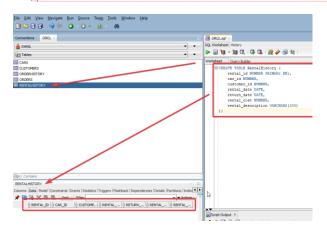
Определяет порядок сортировки результатов по столбцу change\_date в убывающем порядке (от самых новых изменений к более старым).

е. сохранение истории сдачи в аренду авто.

Для реализации сохранения истории сдачи автомобилей в аренду мы создадим таблицу «RentalHistory» и создадим триггер для записи изменений в эту таблицу:

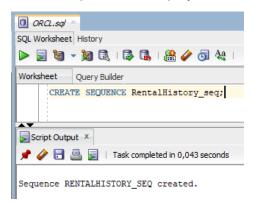
Создание таблицы для истории аренды:

```
CREATE TABLE RentalHistory (
rental_id NUMBER PRIMARY KEY,
car_id NUMBER,
customer_id NUMBER,
rental_date DATE,
return_date DATE,
rental_cost NUMBER,
rental_description VARCHAR2(255)
```



Создание последовательности «RentalHistory\_seq», ее мы будем использовать для генерации уникальных идентификаторов:

CREATE SEQUENCE RentalHistory\_seq;



Создадим триггер для записи сдачи аренды авто:

CREATE OR REPLACE TRIGGER RentalHistoryTrigger AFTER INSERT ON Orders FOR EACH ROW BEGIN Добавлено примечание ([a14]): rental\_id: Это уникальный идентификатор для каждой записи в таблице "RentalHistory". Используется для идентификации конкретной аренды.

car\_id: Этот столбец содержит идентификатор автомобиля, который сдается в аренду.

customer\_id: Здесь хранится идентификатор клиента, сдающего автомобиль в аренду.

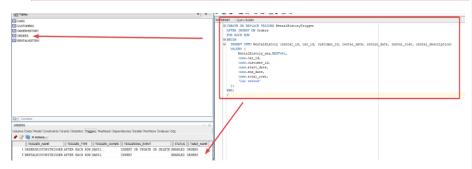
rental\_date: Этот столбец представляет дату начала аренды автомобиля.

return\_date: Здесь записывается дата окончания аренды.

rental\_cost: В этом столбце можно сохранить информацию о стоимости аренды автомобиля.

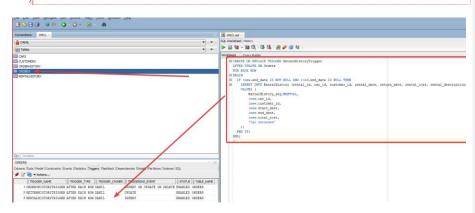
rental\_description: Этот столбец позволяет добавить описание или комментарий к событию сдачи в аренду.

```
INSERT INTO RentalHistory (rental_id, car_id, customer_id, rental_date, return_date, rental_cost, rental_description)
VALUES (
RentalHistory_seq.NEXTVAL,
:new.car_id,
:new.customer_id,
:new.start_date,
:new.end_date,
:new.total_cost,
'Car rented'
);
END;
```



### Создадим триггер для записи возврата аренды авто:

```
CREATE OR REPLACE TRIGGER ReturnHistoryTrigger
AFTER UPDATE ON Orders
FOR EACH ROW
BEGIN
 IF :new.end_date IS NOT NULL AND :old.end_date IS NULL THEN
  INSERT\ INTO\ Rental History\ (rental\_id,\ car\_id,\ customer\_id,\ rental\_date,\ return\_date,\ rental\_cost,
rental description)
  VALUES (
    RentalHistory_seq.NEXTVAL,
     :new.car id.
     :new.customer_id,
    :new.start_date,
    :new.end date,
     :new.total_cost,
     'Car returned'
 END IF;
END;
```



Добавлено примечание ([a15]): CREATE OR REPLACE TRIGGER: Это ключевое слово, которое начинает создание триггера. CREATE создает новый триггер, а OR REPLACE позволяет заменить существующий триггер с тем же именем, если он уже существует.

RentalHistoryTrigger: Это имя триггера.

AFTER INSERT ON Orders: Это событие, на которое триггер будет реагировать. В данном случае, триггер сработает после вставки новой записи в таблицу "Orders".

FOR EACH ROW: Это указывает, что триггер будет выполняться для каждой вставленной строки (записи) в таблице "Orders".

BEGIN...END: Здесь начинается тело триггера, где определяется его действие.

INSERT INTO RentalHistory...: В этой части определено, что при каждой вставке в "Orders", триггер создаст новую запись в таблице "RentalHistory" с информацией о сдаче автомобиля в аренду.

RentalHistory\_seq.NEXTVAL: Это выражение генерирует уникальное значение для столбца "rental\_id" в "RentalHistory".

:new.car\_id, :new.customer\_id: Это значения, взятые из вставленной строки в таблицу "Orders". Они связываются с соответствующими столбцами в "RentalHistory".

:new.start\_date, :new.end\_date, :new.total\_cost: Аналогично, это данные из вставленной строки в "Orders".

Добавлено примечание ([a16]): CREATE OR REPLACE TRIGGER ReturnHistoryTrigger: Эта часть определяет создание нового триггера с именем "ReturnHistoryTrigger" или его замену, если он уже существует с таким именем.

AFTER UPDATE ON Orders: Это определяет событие, при котором триггер будет срабатывать. В данном случае, триггер срабатывает после выполнения операции обновления (UPDATE) в таблице "Orders".

FOR EACH ROW: Это указывает, что триггер выполняется для каждой строки, обновленной в таблице "Orders".

BEGIN ... END;: Здесь располагается основная логика триггера, включая условия и действия, которые будут выполнены.

IF :new.end\_date IS NOT NULL AND :old.end\_date IS NULL THEN: Это условие, которое проверяет, что новое значение поля "end\_date" (:new.end\_date) не является NULL, а старое значение (:old.end\_date) было NULL. Если это условие выполняется, это означает, что заказ завершается (машина возвращается).

INSERT INTO RentalHistory ...: Это действие триггера. Если условие из предыдущего шага истинно, то выполняется вставка новой записи в таблицу "RentalHistory" с информацией о возвращении машины в аренду.

RentalHistory\_seq.NEXTVAL: Это функция, используемая для генерации нового уникального значения для столбца "rental\_id".

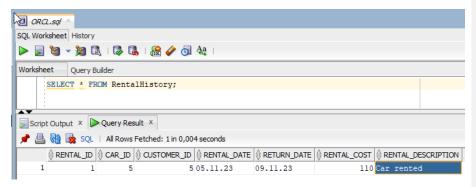
:new.car\_id, :new.customer\_id, :new.start\_date, :new.end\_date, :new.total\_cost: Это значения, взятые из обновленной строки в таблице "Orders".

Чтобы проверить сохранение истории сдачи в аренду авто, создадим тестовую ситуацию, выполним команду:

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (6, 5, 5, TO\_DATE('2023-11-05', 'YYYY-MM-DD'), TO\_DATE('2023-11-09', 'YYYY-MM-DD'), 100.00);

Для проверки сохранения истории сдачи в аренду автомобиля выполним SQL-запрос, который вернет записи из таблицы «RentalHistory»:

SELECT \* FROM RentalHistory;



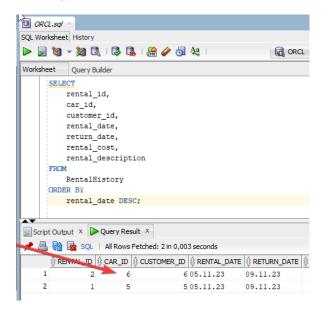
f. отчет об истории сдачи в аренду авто.

## Выполним запрос:

```
SELECT
rental_id,
car_id,
customer_id,
rental_date,
return_date,
rental_cost,
rental_description
FROM
RentalHistory
ORDER BY
rental_date DESC;
```

Чтобы проверить работу отчета об истории сдачи в аренду авто добавим информацию об аренде:

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (7, 6, 6, TO\_DATE('2023-11-05', 'YYYY-MM-DD'), TO\_DATE('2023-11-08', 'YYYY-MM-DD'), 120.00);



Добавлено примечание ([a17]): Этот запрос выбирает следующие данные из таблицы "RentalHistory" и упорядочивает их по дате аренды (сначала новые записи):

rental\_id: Уникальный идентификатор аренды. car\_id: Идентификатор автомобиля, который арендован. customer\_id: Идентификатор клиента, который арендовал автомобиль.

rental\_date: Дата начала аренды. return\_date: Дата возврата автомобиля. rental\_cost: Стоимость аренды.

rental\_description: Описание события аренды.

g. отчет о финансовой деятельности сервиса за период (для простоты будем считать, что стоимость сдачи в аренду на сутки одного автомобиля константна).

Для создания отчета о финансовой деятельности сервиса сдачи автомобилей в аренду за определенный период будем использовать SQL-запрос, который учитывает стоимость аренды на сутки и агрегирует данные из таблицы «RentalHistory».

#### Подготовка данных:

Прежде чем создавать отчет, должна быть таблица «RentalHistory» с данными о сдаче в аренду автомобилей.

### SQL-запрос:

```
SELECT
TO_CHAR(rental_date, 'YYYY-MM') AS rental_month,
SUM(rental_cost) AS total_revenue
FROM
RentalHistory
WHERE
rental_date >= TO_DATE('2023-11-01', 'YYYY-MM-DD')
AND rental_date < TO_DATE('2023-12-01', 'YYYY-MM-DD')
GROUP BY
TO_CHAR(rental_date, 'YYYY-MM')
ORDER BY
rental_month;
```

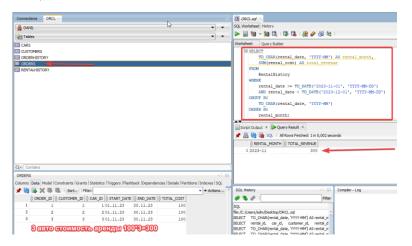
Для проверки работы SQL-запроса удалю все записи из таблиц Orders, OderHistory, Rentalhistory.

После чего внесу данные о заказах снова:

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (1,1, 1, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)
VALUES (2,2, 2, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'),
100.00);

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost) VALUES (3,3, 3, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);



Добавлено примечание ([a18]): Данный SQL-запрос выполняет агрегацию данных из таблицы RentalHistory для создания отчета о доходах (в данном случае за один месяц). Отчет будет включать общий доход для каждого месяца.

SELECT TO \_CHAR(rental\_date, 'YYYY-MM') AS rental\_month: В этой части выбираются данные и преобразуется столбец rental\_date в формат 'YYYY-MM', что представляет собой год и месяц аренды.

SUM(rental\_cost) AS total\_revenue: Эта часть запроса выполняет агрегацию данных. Она суммирует стоимость аренды (rental\_cost) для каждого месяца и предоставляет результат в столбце total\_revenue.

FROM RentalHistory: Указывает источник данных, из которого будут извлекаться записи. В данном случае, это таблица RentalHistory.

WHERE rental\_date >= TO\_DATE('2023-01-01', 'YYYY-MM-DD') AND rental\_date < TO\_DATE('2023-02-01', 'YYYY-MM-DD')

GROUP BY TO\_CHAR(rental\_date, 'YYYY-MM'): Здесь данные группируются по столбцу rental\_date после его преобразования в формат 'YYYY-MM'. Это позволяет сгруппировать данные по месяцам.

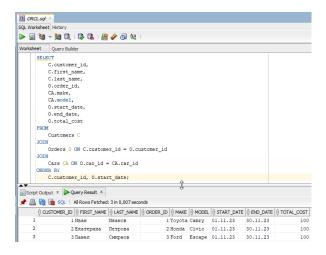
ORDER BY rental\_month: Записи сортируются в результате по месяцам (в формате 'YYYY-MM') в порядке возрастания.

h. Отчет о клиентах: когда какие авто брал в аренду, сколько заплатил и т.д.

Для создания отчета о клиентах, их заказах, арендованных автомобилях и общих расходах мы будем использовать SQL-запрос:

```
SELECT

C.customer_id,
C.first_name,
C.last_name,
O.order_id,
CA.make,
CA.model,
O.start_date,
O.end_date,
O.total_cost
FROM
Customers C
JOIN
Orders O ON C.customer_id = O.customer_id
JOIN
Cars CA ON O.car_id = CA.car_id
ORDER BY
C.customer_id, O.start_date;
```



**Добавлено примечание ([а19]):** SELECT - начинаем запрос и выбираем столбцы, которые будут включены в результат:

C.customer id - идентификатор клиента.

C.first\_name - имя клиента.

C.last\_name - фамилия клиента.

O.order\_id - идентификатор заказа.

CA.make - марка автомобиля.

CA.model - модель автомобиля.

O.start\_date - дата начала аренды.

O.start\_date - дата начала аренды.
O.end\_date - дата окончания аренды.

O.total\_cost - общая стоимость заказа.

FROM - указываем таблицы, из которых будут выбраны

данные:

Customers C - таблица клиентов.

Orders O - таблица заказов. Cars CA - таблица автомобилей.

JOIN - объединяем таблицы на основе связей между

ними:

ON C.customer\_id = O.customer\_id - связываем таблицы Customers и Orders по идентификатору клиента. ON O.car\_id = CA.car\_id - связываем таблицы Orders и Cars по идентификатору арендованного автомобиля. ORDER BY - задаем порядок сортировки результатов:

C.customer\_id - сортируем результаты по идентификатору

клиента.

O.start\_date - затем по дате начала аренды.

 Отчет о текущем состоянии автопарка: сколько в аренде, сколько свободны, сколько на ТО и т.д.

Для создания отчета о клиентах, их заказах, арендованных автомобилях и общих расходах мы будем использовать SQL-запрос:

```
SELECT
COUNT(CASE WHEN status = '6 apende' THEN 1 ELSE NULL END) AS cars_in_rent,
COUNT(CASE WHEN status = '66060den' THEN 1 ELSE NULL END) AS cars_available,
COUNT(CASE WHEN status = 'na TO' THEN 1 ELSE NULL END) AS cars_under_maintenance
FROM Cars;
```

На данном этапе была установлена проблема, что после добавления информации в таблицу Orders, у нас не обновляется информация в таблице Cars (не меняется статус), для обновления мы будем использовать тритгер:

```
CREATE OR REPLACE TRIGGER UpdateCarStatusTrigger
AFTER INSERT OR UPDATE OR DELETE ON Orders
FOR EACH ROW
BEGIN
IF INSERTING THEN
UPDATE Cars
SET status = 'e apende'
WHERE car_id = :new.car_id;
ELSIF DELETING THEN
UPDATE Cars
SET status = 'oocmyneh'
WHERE car_id = :old.car_id;
END IF;
END I;
```

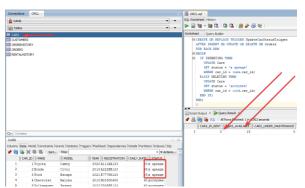
Для проверки работы SQL-запроса удалю все записи из таблиц Orders, OderHistory, Rentalhistory.

После чего внесу данные о заказах снова:

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)
VALUES (1,1, 1, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'),
100.00);

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)
VALUES (2,2, 2, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'),
100.00);

INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)
VALUES (3,3, 3, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'),
100.00);



Добавлено примечание ([a20]): В этом запросе мы используем функцию COUNT с условными выражениями CASE WHEN, чтобы подсчитать количество автомобилей в каждой категории (в аренде, свободных, на TO). Результат запроса будет содержать три столбца: "cars\_in\_rent" (автомобили в аренде), "cars\_available" (свободные автомобили) и "cars\_under\_maintenance" (автомобили на TO).

Добавлено примечание ([a21]): Данный SQL скрипт создает триггер с именем "UpdateCarStatusTrigger", который автоматически обновляет статус автомобиля в таблице "Cars" при вставке, обновлении или удалении данных в таблице "Orders".

CREATE OR REPLACE TRIGGER UpdateCarStatusTrigger: Это начало создания триггера с именем "UpdateCarStatusTrigger".

AFTER INSERT OR UPDATE OR DELETE ON Orders: Триггер срабатывает после операций вставки, обновления и удаления в таблице "Orders".

FOR EACH ROW: Этот триггер будет выполняться для каждой строки, которая затрагивается операцией INSERT, UPDATE или DELETE.

BEGIN: Начало тела триггера.

IF INSERTING THEN: Эта часть проверяет, срабатывает ли триггер после операции вставки (INSERT). Если условие выполняется, выполняются следующие действия:

UPDATE Cars SET status = 'в аренде' WHERE car\_id = :new.car\_id;: Здесь выполняется SQL-запрос для обновления статуса автомобиля в таблице "Cars" на "в аренде" для автомобиля, связанного с данным заказом (определенным как :new.car id).

ELSIF DELETING THEN: Если триггер срабатывает после операции удаления (DELETE), выполняются следующие лействия:

UPDATE Cars SET status = 'свободен' WHERE car\_id = :old.car\_id;: Здесь выполняется SQL-запрос для обновления статуса автомобиля в таблице "Cars" на "доступен" для автомобиля, который был арендован в заказе, который удаляется (определенный как :old.car\_id).

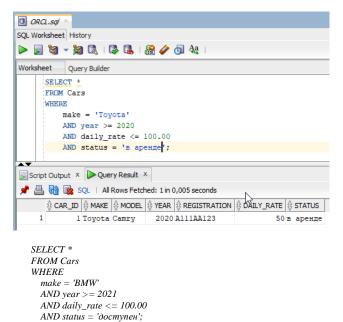
END IF;: Завершение условной конструкции.

END;: Завершение тела триггера.

## ј. Подбор(поиск) авто по параметрам

Для реализации поиска автомобилей по параметрам воспользуемся SQL-запросами и WHERE-условиями, чтобы фильтровать данные в таблице "Cars" в соответствии с заданными параметрами:

```
SELECT *
FROM Cars
WHERE
make = 'Toyota'
AND year >= 2020
AND daity_rate <= 100.00
AND status = '6 apeude';
```



```
SQL Worksheet History

SQL Worksheet Query Builder

SELECT *
FROM Cars
WHERE

make = 'BMW'
AND year >= 2021
AND daily_rate <= 100.00
AND status = 'доступен';

SCript Output x Query Result x

CAR_ID  MAKE  MODEL  YEAR  REGISTRATION DAILY_RATE  STATUS
1 11 BMW X5 2021 P8887Y123 70 доступен
```

Добавлено примечание ([a22]): SELECT \* выбирает все столбцы из таблицы "Cars". Мы можем указать только те столбцы, которые нас интересуют, вместо \*, если нам не нужна вся информация.

FROM Cars указывает таблицу, из которой нужно извлечь данные.

WHERE начинает блок условий для фильтрации результатов.

Каждая строка в блоке WHERE представляет собой условие для фильтрации. В приведенном примере показаны условия фильтрации по марке, году выпуска, дневной ставке аренды и статусу.