

Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное  
учреждение высшего образования  
«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

**ЛАБОРАТОРНАЯ РАБОТА №2**  
**«Запросы на выборку и**  
**модификацию данных,**  
**представления и индексы в**  
**PostgreSQL»**

**Выполнила:**  
студент : Аль-Мошки Исмаил  
Абдулвахаб  
группа: К32401

**Проверили:**  
Говорова Марина Михайловна

Санкт-Петербург  
2023

**Цель работы:** овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

**Оборудование:** компьютерный класс.

**Программное обеспечение:** СУБД PostgreSQL, pgadmin 4.

**Практическое задание:**

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).

### **Вариант 12. БД «Прокат автомобилей»**

Описание предметной области: Компания предоставляет прокат автомобилей. В пункт проката обращаются клиенты, данные о которых регистрируют в базе. Цена проката зависит от марки автомобиля, технических характеристик и года выпуска.

Для проката авто с клиентом заключается договор, в котором фиксируется период проката, вид страховки, стоимость страховки, залоговая стоимость. Залоговая стоимость возвращается полностью или частично клиенту, в зависимости от страховки, аварий и штрафов. Если залоговая стоимость уже возвращена клиенту, но на авто в компанию пришел штраф, то он оплачивается компанией, а не клиентом. При передаче авто клиенту составляется акт о передаче автомобиля клиенту. При возвращении автомобиля также составляется акт о передаче авто компании.

Если клиент не вернул автомобиль в срок и не оформил продление, ему назначается штраф за каждый час просрочки.

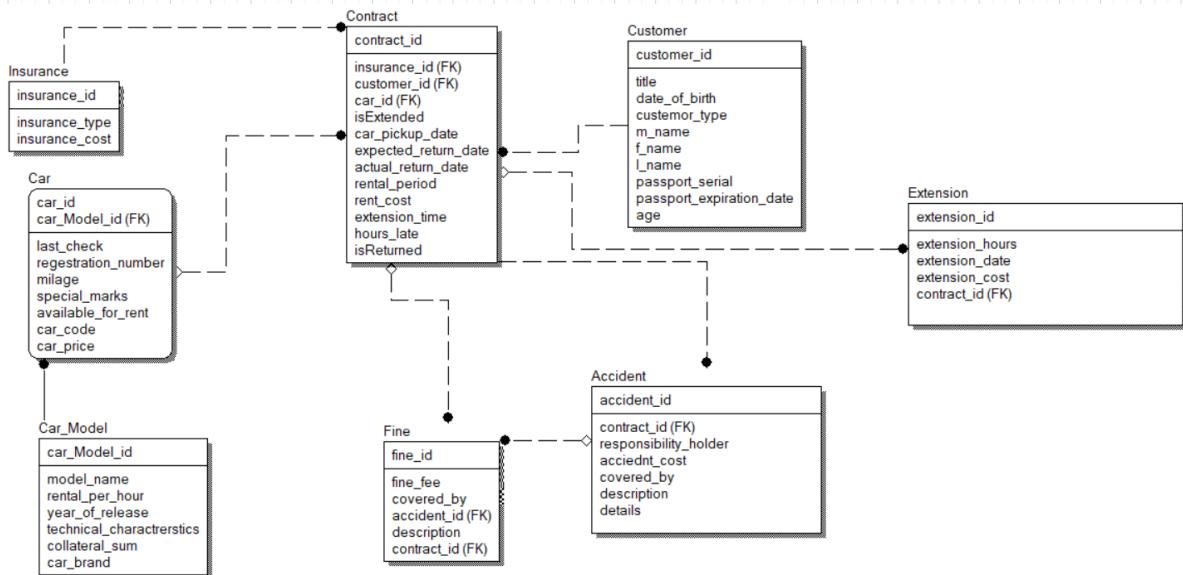
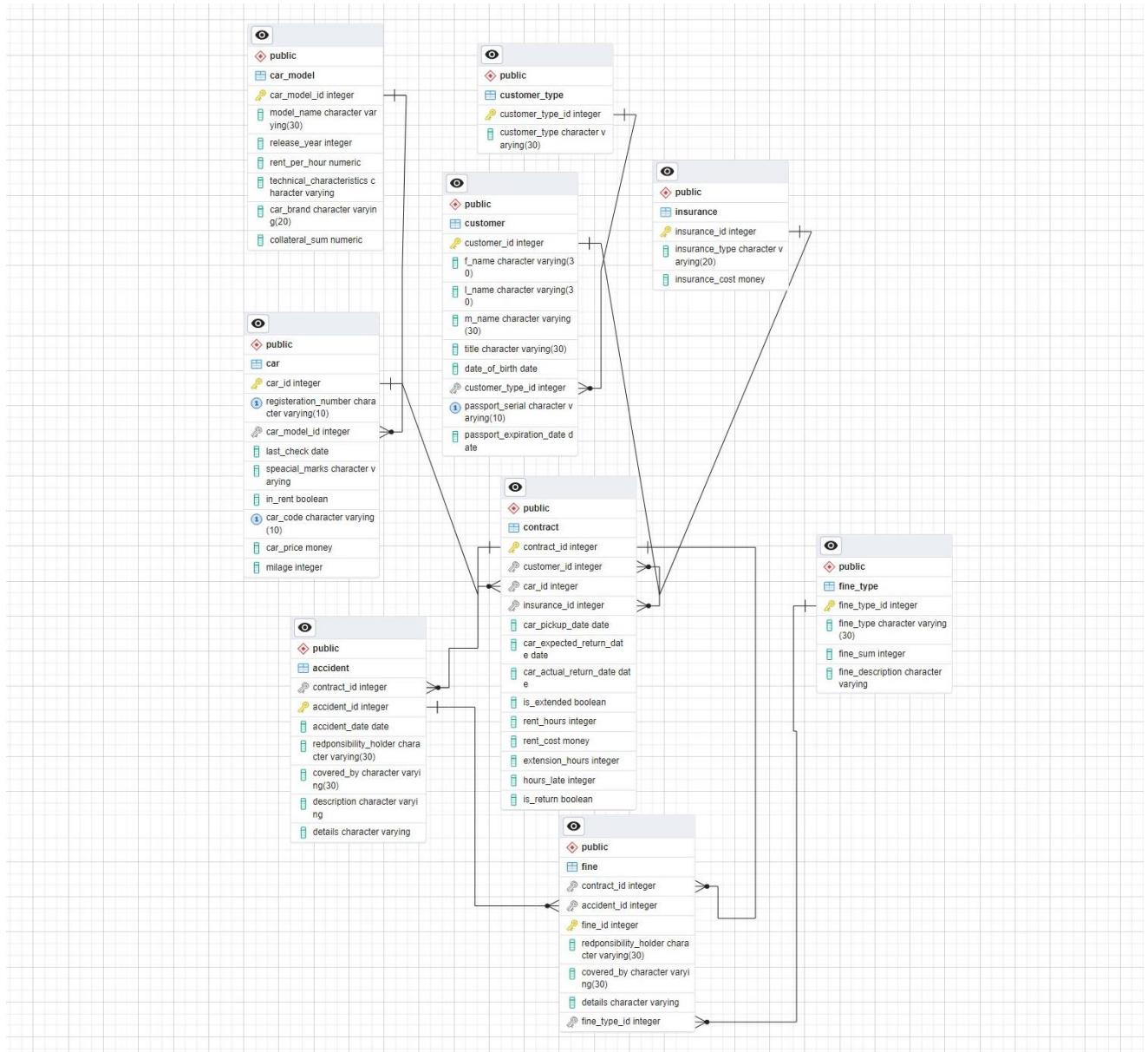
Постоянным клиентам предоставляются скидки.

В системе необходимо хранить историю штрафов и аварий автомобилей.

Цены на прокат автомобилей могут меняться.

БД должна содержать следующий минимальный набор сведений: ФИО. Паспортные данные. Код должности. Наименование должности. Оклад. Обязанности. Код марки. Наименование. Технические характеристики. Описание. Код автомобиля. Регистрационный номер. Номер кузова. Номер двигателя. Год выпуска. Пробег. Цена автомобиля. Цена проката. Дата последнего ТО. Специальные отметки. Отметка о возврате. Код клиента. ФИО. Адрес. Телефон. Паспортные данные. Дата и время выдачи автомобиля. На сколько часов. Дата и время возврата автомобиля. Данные о нарушениях. Данные об авариях. Дата продления. Часов продления.

**Схема базы данных:**



**Задание 2.** Создать запросы:

- Какой автомобиль находился в прокате максимальное количество часов?
- SELECT car\_id from contract JOIN car USING(car\_id) GROUP BY car\_id HAVING sum(rent\_hours) =

```
(SELECT max(sum) FROM
(SELECT sum(rent_hours) FROM contract JOIN car using(car_id) GROUP BY
car_id) as temp1)
```

The screenshot shows the pgAdmin 4 interface. At the top, there's a navigation bar with tabs for Properties, Dashboard, SQL, Statistics, Dependencies, Dependents, Processes, and two database connections: 'Car Rental/postgres@PostgreSQL 15' and 'Untitled\*'. Below the navigation bar is a toolbar with various icons for file operations, search, and database management.

The main area has three tabs: Messages, Query, and Notifications. The 'Query' tab is selected, displaying the following SQL code:

```
1 SELECT car_id from contract JOIN car USING(car_id) GROUP BY car_id
2 HAVING sum (rent_hours) =
3 (SELECT max(sum) FROM
4 (SELECT sum(rent_hours) FROM contract JOIN car using(car_id) GROUP BY car_id) as temp1)
5
```

Below the Query tab is a 'Data Output' tab, which is currently active. It shows a table with one row of data:

car_id	integer
1	4

- Автомобили какой марки чаще всего брались в прокат?
- SELECT car\_id from contract JOIN car USING(car\_id) GROUP BY car\_id  
HAVING count(rent\_hours) =

```
(SELECT max(count) FROM (SELECT sum(rent_hours) FROM contract JOIN car using(car_id)
GROUP BY car_id) as temp1)
```

The screenshot shows a PostgreSQL database interface with two main panels. The top panel is the 'Query' editor, which contains the following SQL code:

```
1 SELECT car_id from contract JOIN car USING(car_id) GROUP BY car_id
2 HAVING count(rent_hours) =
3 (SELECT max(hr) FROM
4 (SELECT count(rent_hours) as hr FROM contract JOIN car using(car_id) GROUP BY car_id) as
```

The bottom panel is the 'Data Output' viewer, which displays the results of the query as a table:

car_id	integer
1	3
2	4

- Определить убытки от простоя автомобилей за вчерашний день.
- ```
SELECT sum(rent_per_hour*24) FROM car
JOIN car_model USING (car_model_id ) WHERE car_id NOT IN
(SELECT car_id FROM contract WHERE car_pickup_date = '2023-5-27' )
```

The screenshot shows the pgAdmin 4 interface. The top bar has tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and two database connections: 'Car Rental/pos...' and 'Car Rental/postgres@Postgr'. The main area has tabs for Messages, Query, Notifications, and Query History. The Query tab contains the following SQL code:

```

1 SELECT sum(rent_per_hour*24) FROM car
2 JOIN car_model USING (car_model_id) WHERE car_id NOT IN
3 (SELECT car_id FROM contract WHERE car_pickup_date = '2023-5-27')

```

The Data Output window below shows the result of the query:

|         | sum   |
|---------|-------|
| numeric |       |
| 1       | 22320 |

- Вывести данные автомобиля, имеющего максимальный пробег.
- SELECT \* FROM car GROUP BY car\_id HAVING milage = (SELECT max(milage) FROM car)**

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects like FTS Dictionaries, Functions, Operators, Procedures, Sequences, Tables, and Contracts. Under 'Tables', 'car\_model' is selected, and its 'Columns' (7) are listed: car\_model\_id, model\_name, release\_year, rent\_per\_hour, technical\_characteristic, car\_brand, and collateral\_sum. The right pane shows the 'Messages' tab with a query window containing the following SQL code:

```

SELECT car_model.car_brand , car_model.model_name, count(*) cnt FROM car_model
JOIN car ON car.car_model_id = car_model.car_model_id
JOIN contract ON contract.car_id = car.car_id
GROUP BY car_model.car_brand , car_model.model_name

```

The results of this query are displayed in a table:

| car_brand | model_name   | cnt |
|-----------|--------------|-----|
| Jeep      | JEEP CHEROKI | 4   |
| Mazda     | TOYOTA       | 3   |

At the bottom of the results window, it says 'Total rows: 2 of 2 Query complete 00:00:00.085 Ln 4, Col 52'.

- Какой автомобиль суммарно находился в прокате дольше всех.
- SELECT \* FROM (SELECT car\_id, sum(rent\_hours) AS hours\_sum FROM contract GROUP BY car\_id )a**

GROUP BY a.car\_id,a.hours\_sum HAVING a.hours\_sum = max(a.hours\_sum)

The screenshot shows the pgAdmin 4 interface. At the top, there's a toolbar with various icons for database management. Below it is a navigation bar with tabs like 'Properties', 'Dashboard', 'SQL', etc. The main area has three panes: 'Messages' (empty), 'Query' (containing the SQL code), and 'Data Output' (showing the result of the query). The 'Query' pane contains the following SQL:

```
1 SELECT * FROM
2 (SELECT car_id, sum(rent_hours) AS hours_sum FROM contract GROUP BY car_id )a
3 GROUP BY a.car_id,a.hours_sum HAVING a.hours_sum = max(a.hours_sum)
4
```

The 'Data Output' pane shows the result of the query:

|   | car_id | hours_sum |
|---|--------|-----------|
| 1 | 3      | 150       |

- Определить, каким количеством автомобилей каждой марки и модели владеет компания.

`SELECT car_brand ,model_name, count(*) FROM car JOIN car_model ON car.car_model_id = car_model.car_model_id GROUP BY car_model.car_brand, car_model.model_name`

The screenshot shows the pgAdmin 4 interface with the database browser on the left and the query editor on the right. The database browser lists tables, columns, constraints, indexes, RLS policies, rules, triggers, and other database objects. The query editor contains the following SQL:

```
SELECT car_brand ,model_name, count(*) FROM car
JOIN car_model ON car.car_model_id = car_model.car_model_id
GROUP BY car_model.car_brand, car_model.model_name
```

The results of the query are displayed in a table:

| car_brand | model_name   | count |
|-----------|--------------|-------|
| 1 Jeep    | Hyundai      | 1     |
| 2 Jeep    | JEEP CHEROKI | 2     |
| 3 Mazda   | TOYOTA       | 1     |
| 4 Mazda   | Mazda car    | 3     |
| 5 Mazda   | Mitsubishi   | 2     |

At the bottom of the query editor, it says 'Total rows: 5 of 5 Query complete 00:00:00.133 Ln 3, Col 1'.

- Определить средний “возраст” автомобилей компании.

```

SELECT model_name, AVG(a.age) FROM (SELECT *, date_part('year', CURRENT_DATE)-release_year AS age FROM car_model
                                     JOIN car ON car.car_model_id = car_model.car_model_id) a GROUP BY model_name

```

The screenshot shows the pgAdmin 4 interface. At the top, there's a navigation bar with links like Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and Car Rental/postgres@Postgr. Below the navigation bar is a toolbar with various icons for file operations and database management. The main area has tabs for Messages, Query, Notifications, and Query History. The Query tab is active, displaying the SQL query shown above. Below the query editor is a "Data Output" viewer. The viewer has a header row with columns labeled "model\_name" and "avg". The data below consists of seven rows of data:

|   | model_name | avg |
|---|------------|-----|
| 1 | Camry      | 5   |
| 2 | Corolla    | 6   |
| 3 | Tucson     | 7   |
| 4 | Accord     | 5   |
| 5 | Highlander | 6   |
| 6 | R8         | 4   |
| 7 | RAV4       | 3   |

**Задание 3.** Создать представление:

- Какой автомобиль ни разу не был в прокате?

CREATE VIEW unused\_cars AS

```

SELECT * FROM car WHERE car_id NOT IN (SELECT car_id FROM contract)

```

Browser    Properties    Dashboard    SQL    Statistics    Dependencies    Dependents    Processes    Car Rental/postg    Untitled\*    Car Rental/postg    < > ×

Servers (1)    PostgreSQL 15    Databases (3)    Car Rental    Casts    Catalogs    Event Triggers    Extensions    Foreign Data Wrappers    Languages    Publications    Schemas (1)    public    Aggregates    Collations    Domains    FTS Configurations    FTS Dictionaries    FTS Parsers    FTS Templates    Foreign Tables    Functions    Materialized Views    Operators    Procedures    Sequences    Tables (9)    accident    car    car\_model    Columns (7)    car\_model\_id

```

CREATE VIEW unused_cars AS
SELECT * FROM car WHERE car_id NOT IN (SELECT car_id FROM contract)

```

Data Output

| car_id | [PK] integer | registration_number | character varying (10) | car_model_id | integer | last_check_date | character varying | speacial_marks | in_rent | boolean | car_code | character varying (10) | car_model_name |
|--------|--------------|---------------------|------------------------|--------------|---------|-----------------|-------------------|----------------|---------|---------|----------|------------------------|----------------|
| 1      |              | 1                   | 12345                  |              |         | 1               | 2021-01-01        | [null]         | false   | run123  | 1        |                        |                |
| 2      |              | 2                   | 12245                  |              |         | 2               | 2020-01-01        | [null]         | false   | rat123  | 1        |                        |                |
| 3      |              | 3                   | 12445                  |              |         | 4               | 2020-01-01        | [null]         | false   | run432  | 1        |                        |                |
| 4      |              | 4                   | 12235                  |              |         | 1               | 2020-01-01        | [null]         | false   | rat654  | 1        |                        |                |
| 5      |              | 5                   | 16656                  |              |         | 5               | 2020-01-01        | [null]         | false   | run875  | 1        |                        |                |
| 6      |              | 6                   | 12647                  |              |         | 2               | 2020-01-01        | [null]         | false   | rat154  | 1        |                        |                |
| 7      |              | 7                   | 24453                  |              |         | 2               | 2020-01-01        | [null]         | false   | run134  | 1        |                        |                |
| 8      |              | 8                   | 23445                  |              |         | 3               | 2020-01-01        | [null]         | false   | rat243  | 1        |                        |                |
| 9      |              | 9                   | 435246                 |              |         | 5               | 2020-01-05        | [null]         | false   | run321  | 1        |                        |                |

Total rows: 9 of 9    Query complete 00:00:00.093    Ln 2, Col 68

- Вывести данные клиентов, не вернувших автомобиль вовремя.

CREATE VIEW late\_returners AS SELECT

DISTINCT customer.customer\_id , customer.f\_name, customer.l\_name, m\_name  
 FROM customer JOIN contract ON contract.customer\_id = customer.customer\_id  
 WHERE (contract.hours\_late IS NOT NULL AND contract.hours\_late > 0)

```

Properties    Dashboard    SQL    Statistics    Dependencies    Dependents    Processes    Car Rental/postg    Untitled*    Car Rental/postg    < > ×

```

Car Rental/postgres@PostgreSQL 15    No limit    < > ×

Messages    Query    Notifications    Query History

```

CREATE VIEW late_returners AS SELECT
DISTINCT customer.customer_id , customer.f_name, customer.l_name, m_name
FROM customer JOIN contract ON contract.customer_id = customer.customer_id
WHERE (contract.hours_late IS NOT NULL AND contract.hours_late > 0)

SELECT * FROM late_returners

```

Data Output

| customer_id | integer | f_name | character varying (30) | l_name  | character varying (30) | m_name   | character varying (30) |
|-------------|---------|--------|------------------------|---------|------------------------|----------|------------------------|
| 1           |         | 4      | MOhameed               | Almoski |                        | ABDO     |                        |
| 2           |         | 1      | Kozman                 | Hesham  |                        | Nasher   |                        |
| 3           |         | 3      | ABDULwahab             | ESMAIL  |                        | Almoshki |                        |

Total rows: 3 of 3    Query complete 00:00:00.092    Ln 7, Col 1

2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.

### 1-INSERT

Update car SET milage = ((SELECT max(milage) FROM car ) +1000) WHERE car\_id = 1

The screenshot shows the pgAdmin 4 interface. At the top, there's a toolbar with various icons for database management. Below it is a navigation bar with tabs like Properties, Dashboard, SQL, Statistics, Dependencies, Dependents, Processes, and Car Rental/pos... The main area has tabs for Messages, Query, Notifications, and Query History. In the Query tab, there are four numbered lines of SQL code:

```
1 SELECT * FROM car
2
3 Update car SET milage = (SELECT max(milage) FROM car ) +1000 WHERE car_model = 3
4
```

Below the query editor is a Data Output viewer. It has tabs for Data Output and Explain. The Data Output tab displays a table with the following data:

|   | car_model_id | last_check_date | speacial_marks | in_rent | car_code | car_price      | milage |
|---|--------------|-----------------|----------------|---------|----------|----------------|--------|
| 1 |              | 1 2021-01-01    | [null]         | false   | run123   | 12,000.00 .₪.j | 6000   |
| 2 |              | 2 2020-01-01    | [null]         | false   | rat123   | 11,000.00 .₪.j | 6000   |
| 3 |              | 4 2020-01-01    | [null]         | false   | run432   | 12,000.00 .₪.j | 6000   |
| 4 |              | 1 2020-01-01    | [null]         | false   | rat654   | 11,500.00 .₪.j | 6000   |
| 5 |              | 5 2020-01-01    | [null]         | false   | run875   | 13,000.00 .₪.j | 6000   |
| 6 |              | 2 2020-01-01    | [null]         | false   | rat154   | 12,000.00 .₪.j | 6000   |
| 7 |              | 2 2020-01-01    | [null]         | false   | run134   | 11,000.00 .₪.j | 6000   |
| 8 |              | 3 2020-01-01    | [null]         | false   | rat243   | 16,000.00 .₪.j | 6000   |

The screenshot shows the pgAdmin 4 interface. The top bar includes tabs for Properties, Dashboard, SQL, Statistics, Dependencies, Dependents, Processes, and several database connections. The main area has tabs for Messages, Query (which is selected), Notifications, and Query History. Below these tabs is a toolbar with icons for file operations, search, and refresh. The Query tab contains the following SQL code:

```

1 Update car SET milage = ((SELECT max(milage) FROM car ) +1000) WHERE car_id = 1
2
3 SELECT * FROM car
4

```

Below the Query tab is a Data Output tab, which displays the results of the last query as a table. The table has columns: car\_model\_id, last\_check\_date, speacial\_marks, in\_rent, car\_code, car\_price, and milage. The data is as follows:

|   | car_model_id | last_check_date | speacial_marks | in_rent | car_code | car_price     | milage |
|---|--------------|-----------------|----------------|---------|----------|---------------|--------|
| 1 | 2            | 2020-01-01      | [null]         | false   | rat123   | 11,000.00 .₪. | 6000   |
| 2 | 4            | 2020-01-01      | [null]         | false   | run432   | 12,000.00 .₪. | 6000   |
| 3 | 1            | 2020-01-01      | [null]         | false   | rat654   | 11,500.00 .₪. | 6000   |
| 4 | 5            | 2020-01-01      | [null]         | false   | run875   | 13,000.00 .₪. | 6000   |
| 5 | 2            | 2020-01-01      | [null]         | false   | rat154   | 12,000.00 .₪. | 6000   |
| 6 | 2            | 2020-01-01      | [null]         | false   | run134   | 11,000.00 .₪. | 6000   |
| 7 | 3            | 2020-01-01      | [null]         | false   | rat243   | 16,000.00 .₪. | 6000   |
| 8 | 1            | 2021-01-01      | [null]         | false   | run123   | 12,000.00 .₪. | 7000   |

## 2-DELETE

DELETE FROM contract

WHERE contract.car\_id IN

(SELECT DISTINCT car.car\_id FROM car JOIN car\_model  
ON car.car\_model\_id = car\_model.car\_model\_id)

WHERE car\_model.release\_year= (SELECT max(release\_year)FROM car\_model) )

Properties Dashboard SQL Statistics Dependencies Dependents Processes Car Rental/pos... Untitled\* Car Rental/pos... < > ↻

Car Rental/postgres@PostgreSQL 15

No limit

Messages Query Notifications Query History

```

1 SELECT * FROM car JOIN car_model
2 USING(car_model_id) JOIN contract USING (car_id)
3
4
5 DELETE FROM contract
6 WHERE contract.car_id IN
7 (SELECT DISTINCT car.car_id FROM car JOIN car_model
8 ON car.car_model_id = car_model.car_model_id
9 WHERE car_model.release_year= (SELECT max(release_year)FROM car_model) )
10
11

```

Data Output Explain ×

|   | car_price     | milage  | model_name             | release_year | rent_per_hour | technical_characteristics | car_b |
|---|---------------|---------|------------------------|--------------|---------------|---------------------------|-------|
|   | money         | integer | character varying (30) | integer      | numeric       | character varying         | chara |
| 1 | 12,000.00 .₪. | 6000    | TOYOTA                 | 2015         | 800           | [null]                    | Mazd  |
| 2 | 11,500.00 .₪. | 6000    | JEEP CHEROKI           | 2019         | 1000          | [null]                    | Jeep  |
| 3 | 12,000.00 .₪. | 6000    | TOYOTA                 | 2015         | 800           | [null]                    | Mazd  |
| 4 | 11,500.00 .₪. | 6000    | JEEP CHEROKI           | 2019         | 1000          | [null]                    | Jeep  |
| 5 | 12,000.00 .₪. | 7000    | JEEP CHEROKI           | 2019         | 1000          | [null]                    | Jeep  |
| 6 | 12,000.00 .₪. | 6000    | TOYOTA                 | 2015         | 800           | [null]                    | Mazd  |
| 7 | 11,500.00 .₪. | 6000    | JEEP CHEROKI           | 2019         | 1000          | [null]                    | Jeep  |

The screenshot shows the pgAdmin interface with the following components:

- Properties**, **Dashboard**, **SQL**, **Statistics**, **Dependencies**, **Dependents**, **Processes**, **Car Rental/pos...**, **Untitled\***, **Car Rental/pos...**, and a back/forward navigation bar.
- Car Rental/postgres@PostgreSQL 15** connection selected in the top-left corner.
- Query History** tab selected in the top-left corner.
- Messages**, **Query**, and **Notifications** tabs in the top-left corner.
- Toolbar** with icons for file operations, search, and various database management functions.
- No limit** dropdown in the toolbar.
- Query Editor** containing the following SQL code:

```
1 SELECT * FROM car JOIN car_model
2 USING(car_model_id) JOIN contract USING (car_id)
3
4
5 DELETE FROM contract
6 WHERE contract.car_id IN
7 (SELECT DISTINCT car.car_id FROM car JOIN car_model
8 ON car.car_model_id = car_model.car_model_id
9 WHERE car_model.release_year= (SELECT max(release_year)FROM car_model) )
10
11
```
- Data Output** tab selected in the bottom-left corner.
- Explain** tab in the bottom-left corner.
- Toolbar** for the Data Output tab with icons for file operations, copy, and export.
- Table** showing the results of the query:

|   | car_price        | milage | model_name | release_year | rent_per_hour | technical_characteristics | ... |
|---|------------------|--------|------------|--------------|---------------|---------------------------|-----|
| 1 | 12,000.00 .ω. .j | 6000   | TOYOTA     | 2015         | 800           | [null]                    | ... |
| 2 | 12,000.00 .ω. .j | 6000   | TOYOTA     | 2015         | 800           | [null]                    | ... |
| 3 | 12,000.00 .ω. .j | 6000   | TOYOTA     | 2015         | 800           | [null]                    | ... |
- Total rows: 3 of 3** at the bottom left.
- Query complete 00:00:00.937** at the bottom left.
- Ln 2, Col 49** at the bottom right.

3- UPDATE

```
UPDATE contract SET car_actual_return_date = CURRENT_DATE WHERE contract_id = 5
```

Properties Dashboard SQL Statistics Dependencies Dependents Processes Car Rental/pos... Untitled\* Car Rental/pos...

Car Rental/postgres@PostgreSQL 15 No limit

Messages Query Notifications Query History

```

1 ALTER TABLE contract ALTER COLUMN rent_cost TYPE INTEGER USING rent_cost::integer
2 UPDATE contract SET rent_cost = CAST((CAST(rent_cost AS decimal)-100) AS MONEY)
3 WHERE car_id IN
4 (SELECT contract.car_id FROM insurance JOIN contract USING (insurance_id) WHERE insurance_type = 'Casual')
5
6
7 SELECT * FROM contract

```

Data Output Explain

|   | car_actual_return_date | is_extended | rent_hours | rent_cost                            | extension_hours | hours_late | is_return |
|---|------------------------|-------------|------------|--------------------------------------|-----------------|------------|-----------|
|   | date                   | boolean     | integer    | money                                | integer         | integer    | boolean   |
| 1 | [null]                 | false       | 70         | 3,900.00 . <sup>,</sup> <sub>,</sub> | [null]          | [null]     | false     |
| 2 | [null]                 | false       | 40         | 4,900.00 . <sup>,</sup> <sub>,</sub> | [null]          | [null]     | false     |
| 3 | [null]                 | false       | 40         | 4,900.00 . <sup>,</sup> <sub>,</sub> | [null]          | 30         | false     |

Properties Dashboard SQL Statistics Dependencies Dependents Processes Car Rental/pos... Untitled\* Car Rental/pos...

Car Rental/postgres@PostgreSQL 15 No limit

Messages Query Notifications Query History

```

1 rent_cost = CAST((CAST(rent_cost AS decimal)-100) AS MONEY)
2
3 id FROM insurance JOIN contract USING (insurance_id) WHERE insurance_type = 'Casual';
4
5 :t

```

Data Output Explain

|   | car_actual_return_date | is_extended | rent_hours | rent_cost                            | extension_hours | hours_late | is_return |
|---|------------------------|-------------|------------|--------------------------------------|-----------------|------------|-----------|
|   | date                   | boolean     | integer    | money                                | integer         | integer    | boolean   |
| 1 | [null]                 | false       | 70         | 3,800.00 . <sup>,</sup> <sub>,</sub> | [null]          | [null]     | false     |
| 2 | [null]                 | false       | 40         | 4,800.00 . <sup>,</sup> <sub>,</sub> | [null]          | [null]     | false     |
| 3 | [null]                 | false       | 40         | 4,800.00 . <sup>,</sup> <sub>,</sub> | [null]          | 30         | false     |

### 3. Изучить графическое представление запросов и просмотреть историю запросов.

Properties Dashboard SQL Statistics Dependencies Dependents Processes Car Rental/pos... Untitled\* Car Rental/pos... < > X

Car Rental/postgres@PostgreSQL 15

No limit

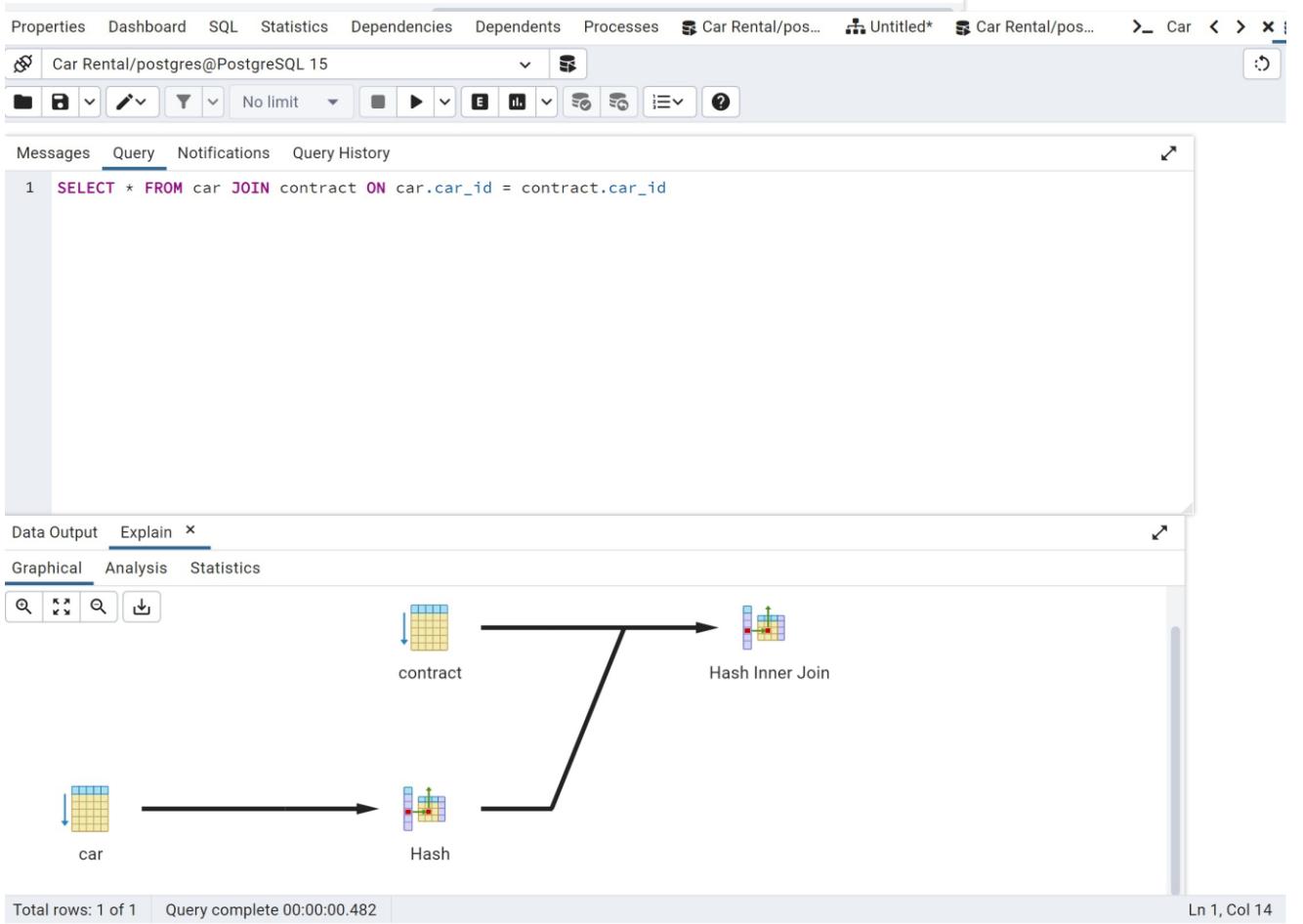
Messages Query Notifications Query History

```
1 UPDATE contract SET rent_cost = rent_cost - 100
2 WHERE car_id IN
3 (SELECT contract.car_id FROM insurance JOIN contract USING (insurance_id) WHERE insura
4
5
6
7 SELECT * FROM contract
```

Data Output Explain

car\_actual\_return\_date is\_extended rent\_hours rent\_cost extension\_hours hours\_late is\_return

|   | date   | boolean | integer | money           | integer | integer | boolean |
|---|--------|---------|---------|-----------------|---------|---------|---------|
| 1 | [null] | false   | 70      | 4,000.00 .ω. .j | [null]  | [null]  | false   |
| 2 | [null] | false   | 40      | 5,000.00 .ω. .j | [null]  | [null]  | false   |
| 3 | [null] | false   | 40      | 5,000.00 .ω. .j | [null]  | 30      | false   |



4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

The screenshot shows the pgAdmin 4 interface. At the top, there's a toolbar with various icons for database management. Below the toolbar, the title bar says "Car Rental/postgres@PostgreSQL 15". The main area has tabs for "Messages", "Query", "Notifications", and "Query History", with "Query" currently selected. A code editor window displays the following SQL query:

```
1 SELECT * FROM customer WHERE f_name = 'KAREEM';
2
```

Below the code editor is a "Data Output" tab, which is also selected. It shows a table with the results of the query. The table has the following columns:

| customer_id [PK] integer | f_name character varying (30) | l_name character varying (30) | m_name character varying (30) | title character varying (30) | date_of_birth date | customer_type_id integer | passport_s character v |
|--------------------------|-------------------------------|-------------------------------|-------------------------------|------------------------------|--------------------|--------------------------|------------------------|
| 1                        | 6                             | KAREEM                        | MOHSEN                        | BO AWF                       | AI expert          | 2001-02-15               | 1 033322534            |

At the bottom of the pgAdmin interface, there are status messages: "Total rows: 1 of 1" and "Query complete 00:00:00.229".

Dashboard Properties SQL Statistics Dependencies Dependents Processes Car Rental/pos... Car Rental/postgres@Postgr < > X

Car Rental/postgres@PostgreSQL 15 No limit

Messages Query Notifications Query History

```
1 CREATE INDEX f_name_l_name_indx
2 ON customer(f_name,l_name);
3 SELECT * FROM customer WHERE (f_name = 'Bob' AND
4 l_name ='Smith');
```

Data Output

| customer_id | f_name | l_name | m_name | title  | date_of_birth | customer_type_id |
|-------------|--------|--------|--------|--------|---------------|------------------|
| 1           | 4      | Bob    | Smith  | [null] | [null]        | 1980-03-03       |

Properties Dashboard SQL Statistics Dependencies Dependents Processes Car Rental/postgres@PostgreSQL 15 Untitled\* Car Rental/postgres@PostgreSQL 15 Car < > X

Car Rental/postgres@PostgreSQL 15 No limit

Messages Query Notifications Query History

```
1 CREATE INDEX f_name_indx ON customer(f_name);
2 SELECT * FROM customer WHERE f_name = 'KAREEM';
```

Data Output Explain

| customer_id | f_name | l_name | m_name | title  | date_of_birth | customer_type_id | passport_s  |
|-------------|--------|--------|--------|--------|---------------|------------------|-------------|
| 1           | 6      | KAREEM | MOHSEN | BO AWF | AI expert     | 2001-02-15       | 1 033322534 |

Total rows: 1 of 1 Query complete 00:00:00.109 Ln 1, Col 46

Индексирование сводит к минимуму время поиска, но оно может отнимать много времени в таблицах, в которые поступает много вставок.

**Вывод:**

В этой лабораторной работе я освоил команды DML SQL в postgresql и научился создавать сложные запросы и подзапросы.