

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Национальный исследовательский университет ИТМО»  
(Университет ИТМО)

ОТЧЕТ

по лабораторной работе № 4

*Проектирование и реализация баз данных*

ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ.  
ПРЕДСТАВЛЕНИЯ. РАБОТА С ИНДЕКСАМИ

Студент:

*Группа № 436209*

*А.А. Цырульников*

Преподаватель:

*Преподаватель практики*

*М.М. Говорова*

Санкт-Петербург 2025

# СОДЕРЖАНИЕ

Стр.

## ВВЕДЕНИЕ

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

### **Практическое задание:**

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию лабораторной работы №2, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.
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.

### **Вариант 14. Индивидуальное задание, часть 2:**

#### **Создать запросы**

1. Вывести данные о водителе, который чаще всех доставляет пассажиров на заданную улицу.
2. Вывести данные об автомобилях, которые имеют пробег более 250 тысяч километров и которые не проходили ТО в текущем году.
3. Сколько раз каждый пассажир воспользовался услугами таксопарка?
4. Вывести данные пассажира, который воспользовался услугами таксопарка максимальное число раз.
5. Вывести данные о водителе, который ездит на самом дорогом автомобиле.
6. Вывести данные пассажира, который всегда ездит с одним и тем же водителем.
7. Какие автомобили имеют пробег больше среднего пробега для своей марки.

### **Создать представление**

- Содержащее сведения о незанятых на данный момент водителях;
- Зарплата всех водителей за вчерашний день.

### **Создать хранимые процедуры**

- Для вывода данных о пассажирах, которые заказывали такси в заданном, как параметр, временном интервале.
- Вывести сведения о том, куда был доставлен пассажир по заданному номеру телефона пассажира.
- Для вычисления суммарного дохода таксопарка за истекший месяц.

### **Создать индексы**

## 1 Ход работы:

Ниже на рисунке 1.1 представлена схема базы данных, которая была создана в ходе выполнения лабораторной работы №3.

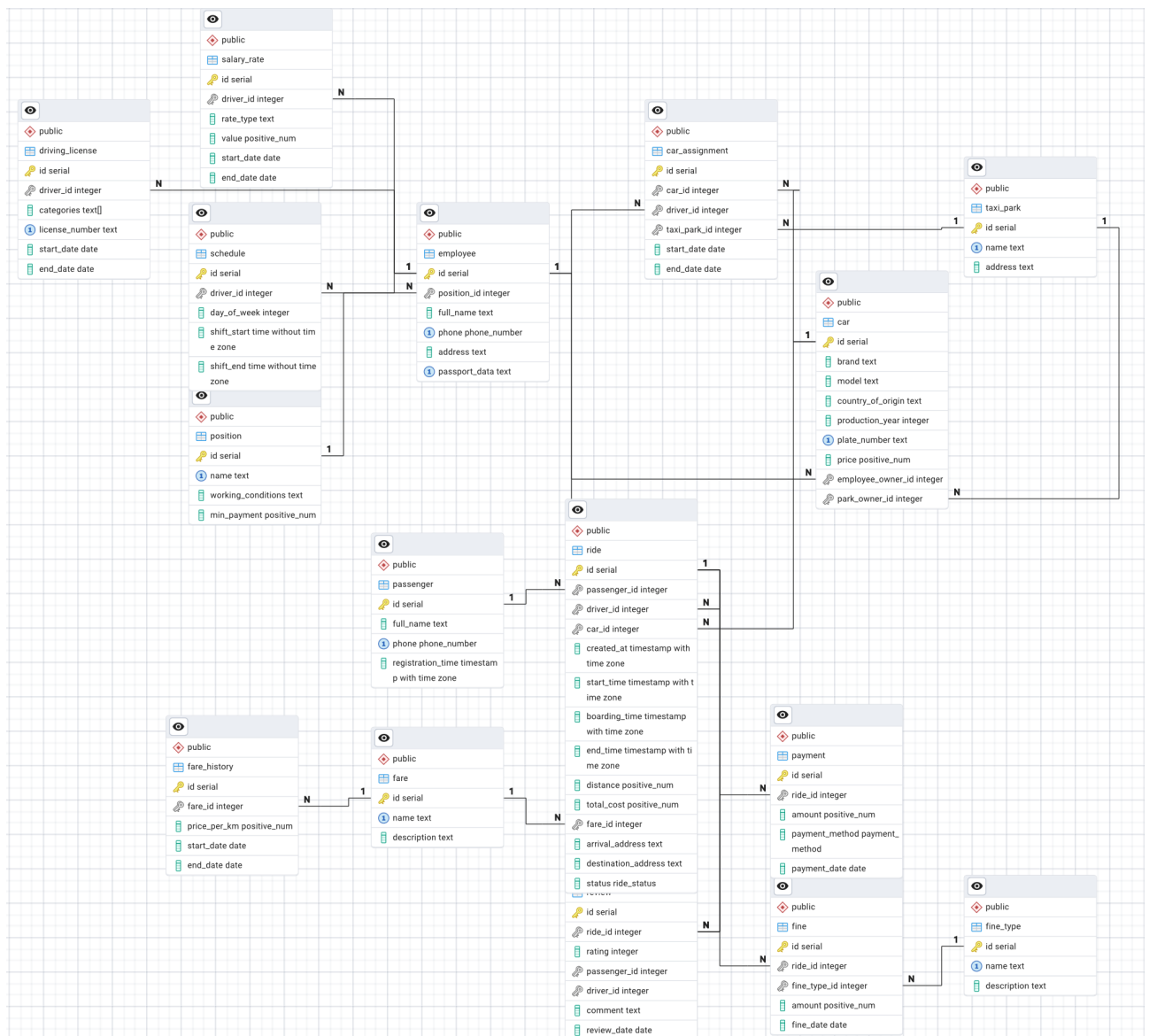


Рисунок 1.1 — Схема базы данных

### 1.1 Запросы на выборку данных

1.1.1 Вывести данные о водителе, который чаще всех доставляет пассажиров на заданную улицу.

```

1  SELECT
2      id,
3      full_name,
4      phone,
5      deliveries_count
6  FROM (
7      SELECT
8          e.id,
9          e.full_name,
10         e.phone,
11         COUNT(*) AS deliveries_count,
12         RANK() OVER (ORDER BY COUNT(*) DESC) AS rnk
13     FROM taxi_service.ride AS r
14     JOIN taxi_service.employee AS e
15         ON e.id = r.driver_id
16     WHERE r.destination_address ILIKE '% ' || '
17           street_name' || '% '
18     GROUP BY e.id, e.full_name, e.phone
19 ) AS ranked
WHERE rnk = 1;

```

	id [PK] integer	full_name text	phone text	deliveries_count bigint
1	6	Исаева Елена Владимировна	+79008889900	1

Рисунок 1.2 — Вывод данных о водителе, который чаще всех доставляет пассажиров на заданную улицу.

**1.1.2 Вывести данные об автомобилях, которые имеют пробег более 250 тысяч километров и которые не проходили ТО в текущем году.**

```

1 SELECT
2 c.id,
3 c.brand,
4 c.model,
5 c.plate_number,
6 cm.mileage
7 FROM taxi_service.car c
8 JOIN taxi_service.car_mileage cm ON c.id = cm.car_id
9 LEFT JOIN (
10 SELECT car_id
11 FROM taxi_service.car_service
12 WHERE EXTRACT(YEAR FROM service_date) = EXTRACT(YEAR
13 FROM CURRENT_DATE)
14 ) cs ON c.id = cs.car_id
15 WHERE cm.mileage > 250000 AND cs.car_id IS NULL;

```

	id integer	brand text	model text	plate_number text	mileage integer
1	3	Kia	Rio	C789YP78	278900
2	6	Ford	Focus	O987TP78	258400

Рисунок 1.3 — Вывод данных об автомобилях, которые имеют пробег более 250 тысяч километров и которые не проходили ТО в текущем году.

### 1.1.3 Сколько раз каждый пассажир воспользовался услугами таксопарка?

```

1 SELECT
2 p.id,
3 p.full_name,
4 p.phone,

```

```

5 COUNT(r.id) AS ride_count
6 FROM taxi_service.passenger AS p
7 LEFT JOIN taxi_service.ride AS r
8 ON p.id = r.passenger_id
9 GROUP BY p.id, p.full_name, p.phone
10 ORDER BY ride_count DESC;

```

	id [PK] integer	full_name text	phone text	ride_count bigint
1	10	Королёва Евгения Александровна	+79005551234	4
2	5	Новиков Алексей Николаевич	+79005556677	3
3	2	Кузнецова Ольга Николаевна	+79002223344	3
4	1	Петров Петр Петрович	+79001112233	3
5	4	Смирнова Анна Владимировна	+79004445566	3
6	8	Киселёв Олег Николаевич	+79003336677	2
7	6	Воробьёв Иван Сергеевич	+79001114455	2
8	3	Морозов Дмитрий Сергеевич	+79003334455	2
9	7	Бабушкина Татьяна Юрьевна	+79002225566	2
10	9	Романова Светлана Петровна	+79004447788	1

Рисунок 1.4 — Вывод данных о количестве поездок каждого пассажира.

#### 1.1.4 Вывести данные пассажира, который воспользовался услугами таксопарка максимальное число раз.

```

1 SELECT
2     id,
3     full_name,
4     phone,
5     ride_count
6 FROM (
7     SELECT
8         p.id,

```



```

9      p.full_name ,
10     p.phone ,
11     COUNT(r.id) AS ride_count ,
12     RANK() OVER (ORDER BY COUNT(r.id) DESC) AS rnk
13 FROM taxi_service.passenger AS p
14 JOIN taxi_service.ride      AS r
15     ON p.id = r.passenger_id
16 GROUP BY p.id, p.full_name , p.phone
17 ) AS ranked
18 WHERE rnk = 1;

```





	id [PK] integer 	full_name text 	phone text 	ride_count bigint 
1	10	Королёва Евгения Александровна	+79005551234	4

Рисунок 1.5 — Вывод данных пассажира, который воспользовался услугами таксопарка максимальное число раз.

### 1.1.5 Вывести данные о водителе, который ездит на самом дорогом автомобиле.

```

1  SELECT
2  id,
3  full_name ,
4  phone ,
5  address ,
6  brand ,
7  model ,
8  price
9 FROM (
10  SELECT
11     e.id,
12     e.full_name ,

```

```

13     e.phone ,
14     e.address ,
15     c.brand ,
16     c.model ,
17     c.price ,
18     RANK() OVER (ORDER BY c.price DESC) AS rnk
19 FROM taxi_service.employee      AS e
20 JOIN taxi_service.car_assignment AS ca
21     ON e.id = ca.driver_id
22 JOIN taxi_service.car           AS c
23     ON ca.car_id = c.id
24 ) AS ranked
25 WHERE rnk = 1;

```

	id integer	full_name text	phone text	address text	brand text	model text	price numeric (10,2)
1	2	Петрова Мария Сергеевна	+79007654321	пр. Мира, д. 15	Toyota	Camry	1200000.00

Рисунок 1.6 — Вывод данных о водителе, который ездит на самом дорогом автомобиле.

### 1.1.6 Вывести данные пассажира, который всегда ездит с одним и тем же водителем.

```

1 SELECT
2 p.id ,
3 p.full_name ,
4 p.phone
5 FROM taxi_service.passenger p
6 WHERE p.id IN (
7 SELECT passenger_id
8 FROM taxi_service.ride
9 GROUP BY passenger_id
10 HAVING COUNT(DISTINCT driver_id) = 1

```

```

11 AND COUNT(*) > 1
12 );

```

	id [PK] integer	full_name text	phone text
1	10	Королёва Евгения Александровна	+79005551234

Рисунок 1.7 — Вывод данных пассажира, который всегда ездит с одним и тем же водителем.

### 1.1.7 Какие автомобили имеют пробег больше среднего пробега для своей марки.

```

1 SELECT
2 c.id,
3 c.brand,
4 c.model,
5 c.plate_number,
6 cm.mileage
7 FROM taxi_service.car c
8 JOIN taxi_service.car_mileage cm ON c.id = cm.car_id
9 JOIN (
10 SELECT
11 c.brand,
12 AVG(cm.mileage) AS avg_mileage
13 FROM taxi_service.car c
14 JOIN taxi_service.car_mileage cm ON c.id = cm.car_id
15 GROUP BY c.brand
16 ) avg_by_brand ON c.brand = avg_by_brand.brand
17 WHERE cm.mileage > avg_by_brand.avg_mileage
18 ORDER BY c.brand, cm.mileage DESC;

```

	id integer 🔒	brand text 🔒	model text 🔒	plate_number text 🔒	mileage integer 🔒
1	9	Hyundai	Elantra	K3330A77	170200
2	1	Hyundai	Solaris	A123BC77	123500
3	3	Kia	Rio	C789YP78	278900
4	10	Toyota	Corolla	P444MH78	210500
5	5	Volkswagen	Polo	M654OP77	186700

Рисунок 1.8 — Вывод данных о автомобилях, которые имеют пробег больше среднего пробега для своей марки.

## 1.2 Запросы на модификацию данных

### 1.2.1 UPDATE: Повышение зарплаты водителям с высоким рейтингом

```

1 UPDATE taxi_service.salary_rate
2 SET value = value * 1.15
3 WHERE driver_id IN (
4     SELECT e.id
5     FROM taxi_service.employee e
6     WHERE e.id IN (
7         SELECT r.driver_id
8         FROM taxi_service.ride r
9         JOIN taxi_service.review rv ON r.id = rv.
            ride_id
10        GROUP BY r.driver_id
11        HAVING AVG(rv.rating) >= 4.5
12    )
13    AND e.position_id = (
14        SELECT id

```

```

15         FROM taxi_service.position
16         WHERE name = 'Driver'
17     )
18 )
19 AND end_date >= CURRENT_DATE;

```

	id integer	full_name text	rate_type text	current_value numeric (10,2)	avg_rating numeric
1	1	Иванов Иван Иванович	Фиксированная	60000.00	5.0000000000000000
2	3	Сидоров Алексей Олегович	По км	15.00	4.6000000000000000

Рисунок 1.9 — Данные до модификации

	id integer	full_name text	rate_type text	new_value numeric (10,2)	avg_rating numeric
1	1	Иванов Иван Иванович	Фиксированная	69000.00	5.0000000000000000
2	3	Сидоров Алексей Олегович	По км	17.25	4.6000000000000000

Рисунок 1.10 — Данные после модификации

### 1.2.2 INSERT: Добавление новых записей о пробеге для автомобилей с высокой активностью

```

1  INSERT INTO car_mileage (car_id, record_date,
2      mileage)
3  SELECT
4      c.id,
5      CURRENT_DATE,
6      COALESCE(
7          (SELECT MAX(mileage) FROM car_mileage WHERE
8              car_id = c.id),
9          100000
10     ) + COALESCE(SUM(r.distance), 0)::INTEGER as
11     new_mileage

```

```

9      FROM car c
10     JOIN ride r ON c.id = r.car_id
11     WHERE r.created_at >= CURRENT_DATE - INTERVAL '7
      days'
12           AND r.status = 'completed'
13           AND c.id NOT IN (
14               SELECT car_id
15               FROM car_mileage
16               WHERE record_date = CURRENT_DATE
17           )
18     GROUP BY c.id
19     HAVING COUNT(r.id) >= 1;

```

	id [PK] integer	brand text	model text	plate_number text	rides_last_7_days bigint	estimated_current_mileage numeric
1	1	Hyundai	Solaris	A123BC77	4	123559.00
2	2	Toyota	Camry	B456EK77	4	95725.00
3	3	Kia	Rio	C789YP78	5	279041.00
4	4	Lada	Granta	H321KM78	2	45321.10
5	5	Volkswagen	Polo	M6540P77	1	186719.60

Рисунок 1.11 — Данные до модификации

	id integer	brand text	model text	plate_number text	record_date date	mileage integer	rides_last_7_days bigint
1	52	Kia	Rio	C789YP78	2025-05-28	279041	5
2	54	Volkswagen	Polo	M6540P77	2025-05-28	186720	1
3	50	Hyundai	Solaris	A123BC77	2025-05-28	123559	4
4	51	Toyota	Camry	B456EK77	2025-05-28	95725	4
5	53	Lada	Granta	H321KM78	2025-05-28	45321	2

Рисунок 1.12 — Данные после модификации

### 1.2.3 DELETE: Удаление старых записей о пробеге для автомобилей с низкой активностью

```
1 DELETE FROM taxi_service.car_mileage
2 WHERE car_id IN (
3     SELECT c.id
4     FROM taxi_service.car c
5     WHERE c.id NOT IN (
6         SELECT DISTINCT car_id
7         FROM taxi_service.ride
8         WHERE created_at >= CURRENT_DATE - INTERVAL '90
9             days'
10     )
11     AND c.production_year < (
12         SELECT AVG(production_year) - 5
13         FROM taxi_service.car
14     )
15 )
16 AND record_date < CURRENT_DATE - INTERVAL '1 year';
```

	id integer 🔒	brand text 🔒	model text 🔒	production_year integer 🔒	record_date date 🔒	mileage integer 🔒	threshold_year numeric 🔒
1	1	Hyundai	Solaris	2020	2024-10-15	123500	2012.3157894736842105
2	2	Toyota	Camry	2021	2024-10-10	95600	2012.3157894736842105
3	3	Kia	Rio	2019	2024-10-12	278900	2012.3157894736842105
4	4	Lada	Granta	2022	2024-10-14	45300	2012.3157894736842105
5	5	Volkswagen	Polo	2020	2024-10-11	186700	2012.3157894736842105
6	6	Ford	Focus	2019	2024-10-13	258400	2012.3157894736842105
7	7	Renault	Logan	2021	2024-10-15	134200	2012.3157894736842105
8	8	Skoda	Octavia	2018	2024-10-10	198300	2012.3157894736842105
9	9	Hyundai	Elantra	2019	2024-10-12	170200	2012.3157894736842105
10	10	Toyota	Corolla	2018	2024-10-14	210500	2012.3157894736842105
11	11	Kia	Sportage	2020	2024-10-15	89400	2012.3157894736842105
12	12	Volkswagen	Tiguan	2021	2024-10-11	65900	2012.3157894736842105
13	13	Hyundai	Creta	2021	2024-10-13	45800	2012.3157894736842105
14	14	Toyota	RAV4	2020	2024-10-10	105300	2012.3157894736842105
15	15	BA3	2107	2010	2022-01-15	250000	2012.3157894736842105
16	16	BA3	2107	2010	2022-06-20	255000	2012.3157894736842105
17	17	BA3	2107	2010	2023-01-10	260000	2012.3157894736842105
18	18	BA3	2107	2010	2023-06-15	265000	2012.3157894736842105
19	19	BA3	2107	2010	2023-10-01	268000	2012.3157894736842105
20	20	BA3	2114	2012	2022-03-10	180000	2012.3157894736842105
21	21	BA3	2114	2012	2022-08-15	185000	2012.3157894736842105
22	22	BA3	2114	2012	2023-02-20	190000	2012.3157894736842105
23	23	BA3	2114	2012	2023-07-25	195000	2012.3157894736842105
24	24	BA3	2114	2012	2023-08-28	197000	2012.3157894736842105

Рисунок 1.13 — Данные до модификации



	id integer	brand text	model text	production_year integer	record_date date	mileage integer	threshold_year numeric
1	1	Hyundai	Solaris	2020	2024-10-15	123500	2012.3157894736842105
2	2	Toyota	Camry	2021	2024-10-10	95600	2012.3157894736842105
3	3	Kia	Rio	2019	2024-10-12	278900	2012.3157894736842105
4	4	Lada	Granta	2022	2024-10-14	45300	2012.3157894736842105
5	5	Volkswagen	Polo	2020	2024-10-11	186700	2012.3157894736842105
6	6	Ford	Focus	2019	2024-10-13	258400	2012.3157894736842105
7	7	Renault	Logan	2021	2024-10-15	134200	2012.3157894736842105
8	8	Skoda	Octavia	2018	2024-10-10	198300	2012.3157894736842105
9	9	Hyundai	Elantra	2019	2024-10-12	170200	2012.3157894736842105
10	10	Toyota	Corolla	2018	2024-10-14	210500	2012.3157894736842105
11	11	Kia	Sportage	2020	2024-10-15	89400	2012.3157894736842105
12	12	Volkswagen	Tiguan	2021	2024-10-11	65900	2012.3157894736842105
13	13	Hyundai	Creta	2021	2024-10-13	45800	2012.3157894736842105
14	14	Toyota	RAV4	2020	2024-10-10	105300	2012.3157894736842105
15	40	BA3	2107	2010	2024-11-01	270000	2012.3157894736842105
16	41	BA3	2114	2012	2024-11-02	200000	2012.3157894736842105
17	42	Daewoo	Nexia	2011	2024-11-03	240000	2012.3157894736842105
18	43	Chevrolet	Lanos	2009	2024-11-04	320000	2012.3157894736842105
19	44	Ford	Sierra	2008	2024-11-05	370000	2012.3157894736842105

Рисунок 1.14 — Данные после модификации

## 1.3 Представления

### 1.3.1 Представление 1: Незанятые водители

```

1 CREATE OR REPLACE VIEW free_drivers AS
2 SELECT
3     e.id,
4     e.full_name,
5     e.phone,
6     e.address,
7     p.name AS position_name,
8     dl.categories,
9     dl.license_number

```

```

10 FROM
11     employee e
12 JOIN
13     position p ON e.position_id = p.id
14 LEFT JOIN
15     driving_license dl ON e.id = dl.driver_id
16 WHERE
17     p.name = 'Driver'
18     AND e.id NOT IN (
19         SELECT DISTINCT driver_id
20         FROM ride
21         WHERE status = 'in_progress'
22     )
23     AND e.id NOT IN (
24         SELECT DISTINCT driver_id
25         FROM car_assignment
26         WHERE CURRENT_DATE BETWEEN start_date AND
27             end_date
28     );

```

	id integer 🔒	full_name text 🔒	phone text 🔒	address text 🔒	position_name text 🔒	categories text[] 🔒	license_number text 🔒
1	4	Смирнов Иван Петрович	+79006667788	ул. Ломоносова, д. 3	Водитель	{B,C}	BZ1999334
2	5	Ковалев Павел Дмитриевич	+79007778899	ул. Гагарина, д. 4	Водитель	{B,C}	BZ1151334
3	6	Исаева Елена Владимировна	+79008889900	пр. Ленина, д. 22	Водитель	{B,C}	BZ1188334
4	9	Николаев Сергей Иванович	+79012345678	ул. Гоголя, д. 5	Водитель	{B}	CD9876543
5	10	Андреева Ольга Викторовна	+79023456789	пр. Невский, д. 120	Водитель	{B,C}	FG5432109
6	11	Соколов Дмитрий Андреевич	+79034567890	ул. Дзержинского, д. 8	Водитель	{B,D}	HJ2468013

Рисунок 1.15 — Представление 1: Незанятые водители

### 1.3.2 Представление 2: Зарплата всех водителей за вчерашний день

```

1 CREATE OR REPLACE VIEW driver_salary_yesterday AS
2 SELECT

```

```

3 e.id,
4 e.full_name,
5 CASE
6 WHEN sr.rate_type = 'Fixed' THEN sr.value / 30
7 WHEN sr.rate_type = 'Hourly' THEN
8 (SELECT COALESCE(SUM(EXTRACT(EPOCH FROM (
9 end_time - start_time))) / 3600), 0) * sr.value
10 FROM ride
11 WHERE driver_id = e.id
12 AND DATE(end_time) = CURRENT_DATE -
13 INTERVAL '1 day'
14 AND status = 'completed')
15 WHEN sr.rate_type = 'Per_km' THEN
16 (SELECT COALESCE(SUM(distance), 0) * sr.value
17 FROM ride
18 WHERE driver_id = e.id
19 AND DATE(end_time) = CURRENT_DATE -
20 INTERVAL '1 day'
21 AND status = 'completed')
22 ELSE 0
23 END AS salary
24 FROM
25 employee e
26 JOIN
27 position p ON e.position_id = p.id
28 LEFT JOIN
29 salary_rate sr ON e.id = sr.driver_id AND
30 CURRENT_DATE - INTERVAL '1 day' BETWEEN sr.
31 start_date AND sr.end_date
32 WHERE
33 p.name = 'Driver';

```

	id [PK] integer 	full_name text 	salary double precision 
1	1	Иванов Иван Иванович	2300
2	2	Петрова Мария Сергеевна	3200
3	3	Сидоров Алексей Олегович	2161.425
4	4	Смирнов Иван Петрович	0
5	5	Ковалев Павел Дмитриевич	0
6	6	Исаева Елена Владимировна	0
7	9	Николаев Сергей Иванович	0
8	10	Андреева Ольга Викторовна	0
9	11	Соколов Дмитрий Андреевич	0

Рисунок 1.16 — Представление 2: Зарплата всех водителей за вчерашний день

## 1.4 Индексы

### 1.4.1 Простой индекс на $\text{passenger}_{id}$

```

1  EXPLAIN (ANALYZE, BUFFERS)
2  SELECT r.id, r.created_at, r.status, r.total_cost,
   r.destination_address
3  FROM ride r
4  WHERE r.passenger_id = 1
5  ORDER BY r.created_at DESC;
```

```

1  CREATE INDEX IF NOT EXISTS idx_ride_passenger_id ON
   ride(passenger_id);
```

	QUERY PLAN text	
1	Sort (cost=188.83..190.10 rows=507 width=53) (actual time=0.795..0.822 rows=507 loops=1)	
2	Sort Key: created_at DESC	
3	Sort Method: quicksort Memory: 96kB	
4	Buffers: shared hit=103	
5	-> Seq Scan on ride r (cost=0.00..166.05 rows=507 width=53) (actual time=0.007..0.705 rows=507 loop...	
6	Filter: (passenger_id = 1)	
7	Rows Removed by Filter: 4537	
8	Buffers: shared hit=103	
9	Planning time: 0.135 ms	
10	Execution time: 0.855 ms	

Рисунок 1.17 — План запроса без индекса

	QUERY PLAN text	
1	Sort (cost=144.33..145.60 rows=507 width=53) (actual time=0.281..0.305 rows=507 loops=1)	
2	Sort Key: created_at DESC	
3	Sort Method: quicksort Memory: 96kB	
4	Buffers: shared hit=103 read=3	
5	-> Bitmap Heap Scan on ride r (cost=12.21..121.55 rows=507 width=53) (actual time=0.036..0.199 rows=507 loops=1)	
6	Recheck Cond: (passenger_id = 1)	
7	Heap Blocks: exact=103	
8	Buffers: shared hit=103 read=3	
9	-> Bitmap Index Scan on idx_ride_passenger_id (cost=0.00..12.09 rows=507 width=0) (actual time=0.024..0.024 rows=507 l...	
10	Index Cond: (passenger_id = 1)	
11	Buffers: shared read=3	
12	Planning time: 0.129 ms	
13	Execution time: 0.339 ms	

Рисунок 1.18 — План запроса с индексом

#### 1.4.2 Составной индекс на $(driver_id, status, created_at)$

1	EXPLAIN (ANALYZE , BUFFERS)
2	SELECT
3	id ,
4	created_at ,

```

5      distance ,
6      total_cost ,
7      EXTRACT(EPOCH FROM end_time - start_time)/3600 AS
        duration_hours
8  FROM ride
9  WHERE driver_id = 1
10     AND status      = 'completed'
11     AND created_at >= CURRENT_DATE - INTERVAL '7 days
12
ORDER BY created_at DESC;

```

```

1  CREATE INDEX IF NOT EXISTS
    idx_ride_driver_status_date ON ride(driver_id,
    status , created_at);

```


	QUERY PLAN	
	text	
1	Sort (cost=217.61..217.69 rows=34 width=30) (actual time=0.438..0.439 rows=36 loops=1)	
2	Sort Key: created_at DESC	
3	Sort Method: quicksort Memory: 27kB	
4	Buffers: shared hit=103	
5	-> Seq Scan on ride r (cost=0.00..216.75 rows=34 width=30) (actual time=0.009..0.430 rows=36 loops=1)	
6	Filter: ((driver_id = 1) AND (status = 'completed':ride_status) AND (created_at >= (CURRENT_DATE - '7 days':inte...	
7	Rows Removed by Filter: 5008	
8	Buffers: shared hit=103	
9	Planning time: 0.143 ms	
10	Execution time: 0.451 ms	

Рисунок 1.19 — План запроса без индекса


	QUERY PLAN	
	text	
1	Sort (cost=78.03..78.12 rows=34 width=30) (actual time=0.079..0.082 rows=36 loops=1)	
2	Sort Key: created_at DESC	
3	Sort Method: quicksort Memory: 27kB	
4	Buffers: shared hit=8 read=2	
5	-> Bitmap Heap Scan on ride r (cost=4.72..77.17 rows=34 width=30) (actual time=0.050..0.067 rows=36 loops=1)	
6	Recheck Cond: ((driver_id = 1) AND (status = 'completed'::ride_status) AND (created_at >= (CURRENT_DATE - '7 days'::i...	
7	Heap Blocks: exact=5	
8	Buffers: shared hit=8 read=2	
9	-> Bitmap Index Scan on idx_ride_driver_status_date (cost=0.00..4.71 rows=34 width=0) (actual time=0.041..0.041 ro...	
10	Index Cond: ((driver_id = 1) AND (status = 'completed'::ride_status) AND (created_at >= (CURRENT_DATE - '7 days'::i...	
11	Buffers: shared hit=3 read=2	
12	Planning time: 0.266 ms	
13	Execution time: 0.116 ms	

Рисунок 1.20 — План запроса с индексом

## ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были освоены навыки создания сложных SQL-запросов в PostgreSQL для получения данных из базы таксопарка. Выполнено создание представлений для упрощения доступа к часто используемой информации, например, о незанятых водителях. Разработаны запросы с группировкой и подзапросами для анализа данных о поездках, водителях и автомобилях. Освоена оптимизация запросов с помощью индексов, что позволяет ускорить выборку данных. Также изучена модификация данных (добавление, обновление, удаление) с использованием подзапросов.