БД «Прокат автомобилей»

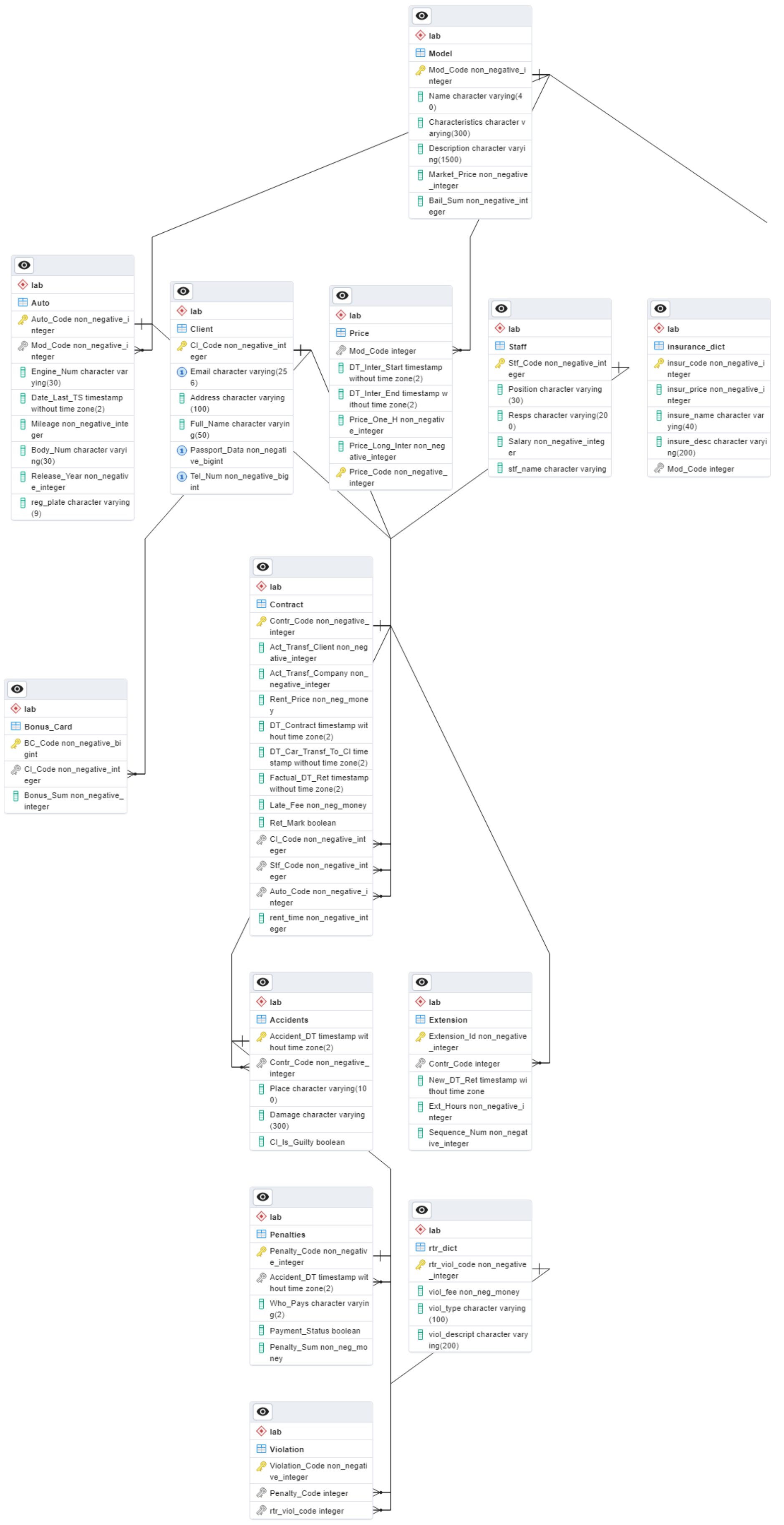
1. Создание запросов и представления на выборку данных к базе данных PostgreSQL

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

3. Графическое представление запросов и история запросов.

4. Простой и составной индексы для двух произвольных запросов и сравнение времени выполнения запросов без индексов и с индексами. Команда EXPLAIN.

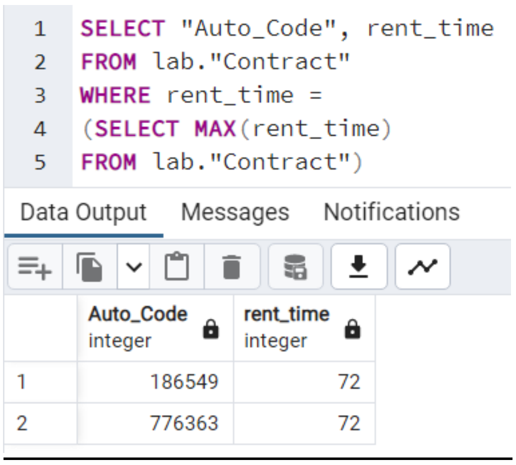
1. ERD Схема:


lab
Model
Mod_Code non_negative_integer
Name character varying(40)
Characteristics character varying(300)
Description character varying(1500)
Market_Price non_negative_integer
Bail_Sum non_negative_integer
lab
Auto
Auto_Code non_negative_integer
Mod_Code non_negative_integer
Engine_Num character varying(30)
Date_Last_TS timestamp without time zone(2)
Mileage non_negative_integer
Body_Num character varying(30)
Release_Year non_negative_integer
reg_plate character varying(9)
lab
Client
Cl_Code non_negative_integer
Email character varying(256)
Address character varying(100)
Full_Name character varying(50)
Passport_Data non_negative_bigint
Tel_Num non_negative_bigint
lab
Price
Mod_Code integer
DT_Inter_Start timestamp without time zone(2)
DT_Inter_End timestamp without time zone(2)
Price_One_H non_negative_integer
Price_Long_Inter non_negative_integer
Price_Code non_negative_integer
lab
Staff
Stf_Code non_negative_integer
Position character varying(30)
Resps character varying(200)
Salary non_negative_integer
stf_name character varying
lab
insurance_dict
insur_code non_negative_integer
insur_price non_negative_integer
insure_name character varying(40)
insure_desc character varying(200)
Mod_Code integer
lab
Contract
Contr_Code non_negative_integer
Act_Transf_Client non_negative_integer
Act_Transf_Company non_negative_integer
Rent_Price non_neg_money
DT_Contract timestamp without time zone(2)
DT_Car_Transf_To_Cl timestamp without time zone(2)
Factual_DT_Ret timestamp without time zone(2)
Late_Fee non_neg_money
Ret_Mark boolean
Cl_Code non_negative_integer
Stf_Code non_negative_integer
Auto_Code non_negative_integer
rent_time non_negative_integer
lab
Bonus_Card
BC_Code non_negative_bigint
Cl_Code non_negative_integer
Bonus_Sum non_negative_integer
lab
Accidents
Accident_DT timestamp without time zone(2)
Contr_Code non_negative_integer
Place character varying(100)
Damage character varying(300)
Cl_Is_Guilty boolean
lab
Extension
Extension_Id non_negative_integer
Contr_Code integer
New_DT_Ret timestamp without time zone
Ext_Hours non_negative_integer
Sequence_Num non_negative_integer
lab
Penalties
Penalty_Code non_negative_integer
Accident_DT timestamp without time zone(2)
Who_Pays character varying(2)
Payment_Status boolean
Penalty_Sum non_neg_money
lab
rtr_dict
rtr_viol_code non_negative_integer
viol_fee non_neg_money
viol_type character varying(100)
viol_descript character varying(200)
lab
Violation
Violation_Code non_negative_integer
Penalty_Code integer
rtr_viol_code integer

2. Выполнение:

Создание запросов:

- Какой автомобиль находился в прокате максимальное количество часов?

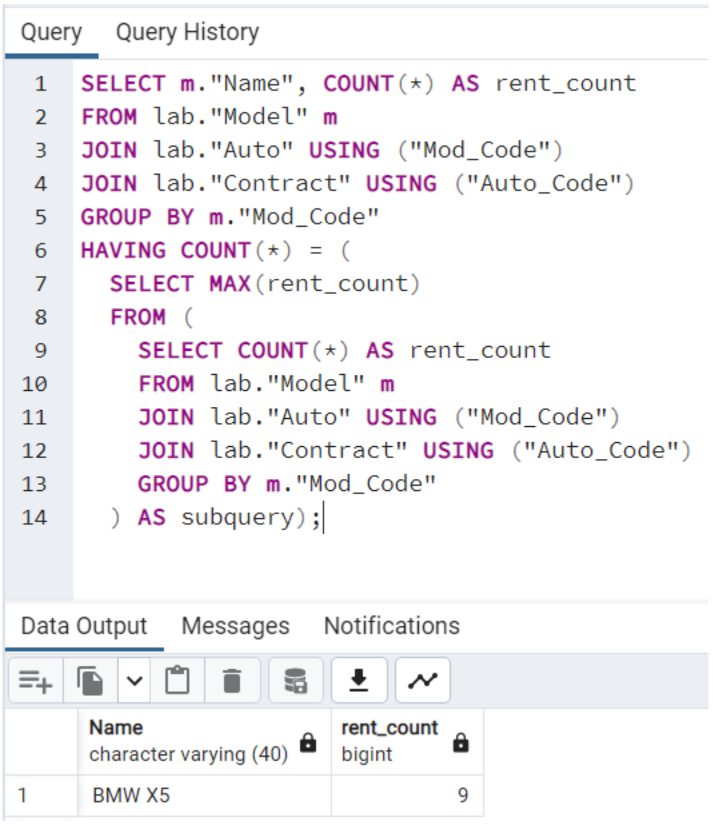


```sql
SELECT "Auto_Code", rent_time
FROM lab."Contract"
WHERE rent_time =
(SELECT MAX(rent_time)
FROM lab."Contract")
```

Data Output   Messages   Notifications

	Auto_Code integer	rent_time integer
1	186549	72
2	776363	72

- Автомобили какой марки чаще всего брались в прокат?



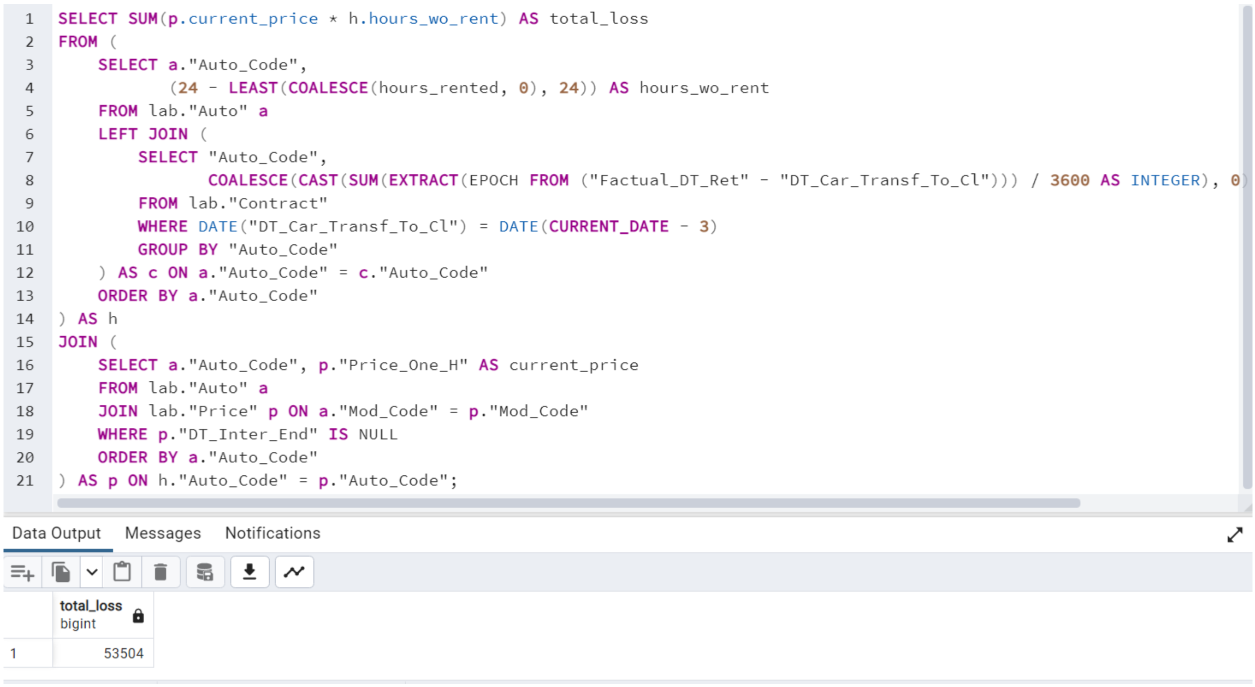
Query   Query History

```sql
SELECT m."Name", COUNT(*) AS rent_count
FROM lab."Model" m
JOIN lab."Auto" USING ("Mod_Code")
JOIN lab."Contract" USING ("Auto_Code")
GROUP BY m."Mod_Code"
HAVING COUNT(*) = (
  SELECT MAX(rent_count)
  FROM (
    SELECT COUNT(*) AS rent_count
    FROM lab."Model" m
    JOIN lab."Auto" USING ("Mod_Code")
    JOIN lab."Contract" USING ("Auto_Code")
    GROUP BY m."Mod_Code"
  ) AS subquery);
```

Data Output   Messages   Notifications

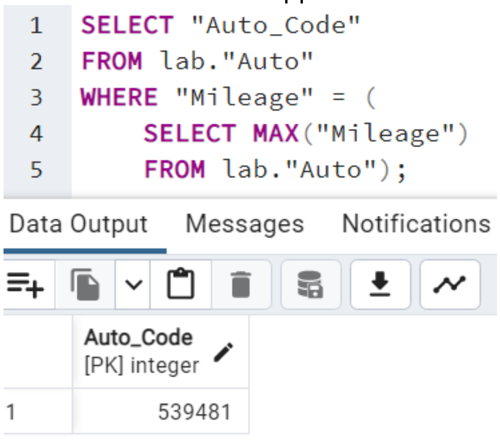
	Name character varying (40)	rent_count bigint
1	BMW X5	9

- Определить убытки от простоя автомобилей за вчерашний день.


```sql
SELECT SUM(p.current_price * h.hours_wo_rent) AS total_loss
FROM (
    SELECT a."Auto_Code",
           (24 - LEAST(COALESCE(hours_rented, 0), 24)) AS hours_wo_rent
    FROM lab."Auto" a
    LEFT JOIN (
        SELECT "Auto_Code",
               COALESCE(CAST(SUM(EXTRACT(EPOCH FROM ("Factual_DT_Ret" - "DT_Car_Transf_To_Cl"))) / 3600 AS INTEGER), 0
        FROM lab."Contract"
        WHERE DATE("DT_Car_Transf_To_Cl") = DATE(CURRENT_DATE - 3)
        GROUP BY "Auto_Code"
    ) AS c ON a."Auto_Code" = c."Auto_Code"
    ORDER BY a."Auto_Code"
) AS h
JOIN (
    SELECT a."Auto_Code", p."Price_One_H" AS current_price
    FROM lab."Auto" a
    JOIN lab."Price" p ON a."Mod_Code" = p."Mod_Code"
    WHERE p."DT_Inter_End" IS NULL
    ORDER BY a."Auto_Code"
) AS p ON h."Auto_Code" = p."Auto_Code";
```

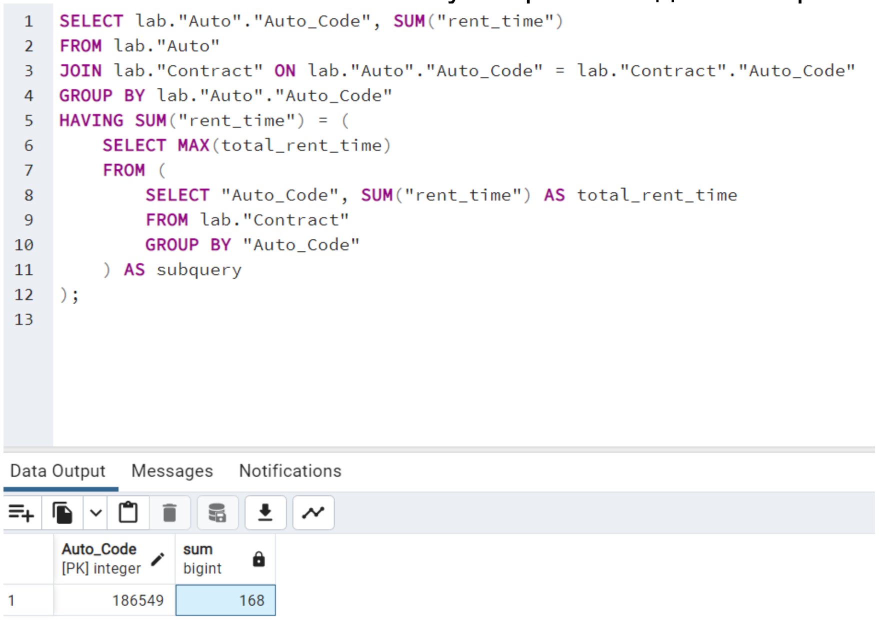
```
   total_loss
   bigint
1       53504
```

- Вывести данные автомобиля, имеющего максимальный пробег.


```sql
SELECT "Auto_Code"
FROM lab."Auto"
WHERE "Mileage" = (
    SELECT MAX("Mileage")
    FROM lab."Auto");
```

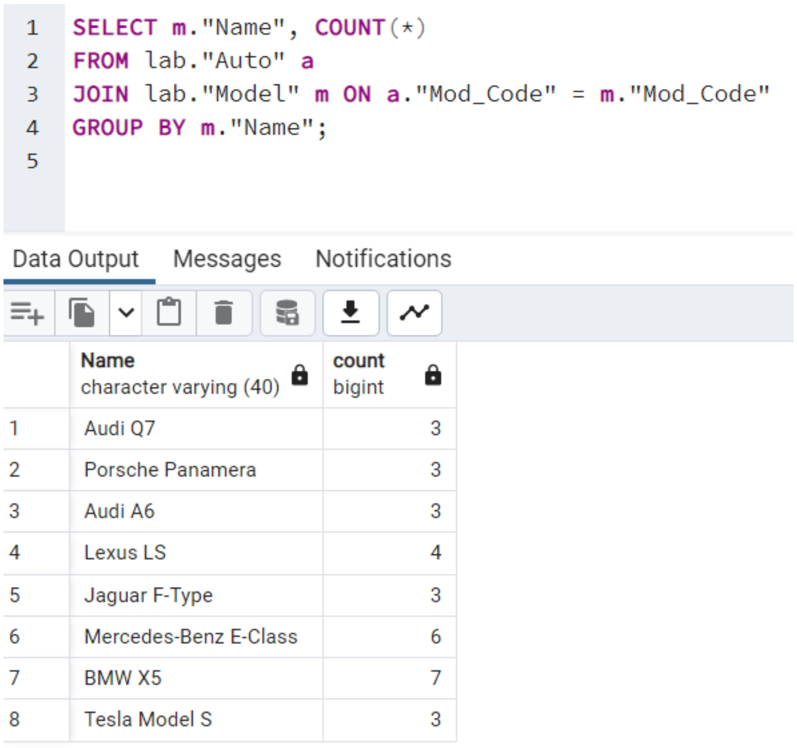
```
   Auto_Code
   [PK] integer
1       539481
```

- Какой автомобиль суммарно находился в прокате дольше всех.


```sql
SELECT lab."Auto"."Auto_Code", SUM("rent_time")
FROM lab."Auto"
JOIN lab."Contract" ON lab."Auto"."Auto_Code" = lab."Contract"."Auto_Code"
GROUP BY lab."Auto"."Auto_Code"
HAVING SUM("rent_time") = (
    SELECT MAX(total_rent_time)
    FROM (
        SELECT "Auto_Code", SUM("rent_time") AS total_rent_time
        FROM lab."Contract"
        GROUP BY "Auto_Code"
    ) AS subquery
);
```

```
   Auto_Code      sum
   [PK] integer   bigint
1       186549       168
```

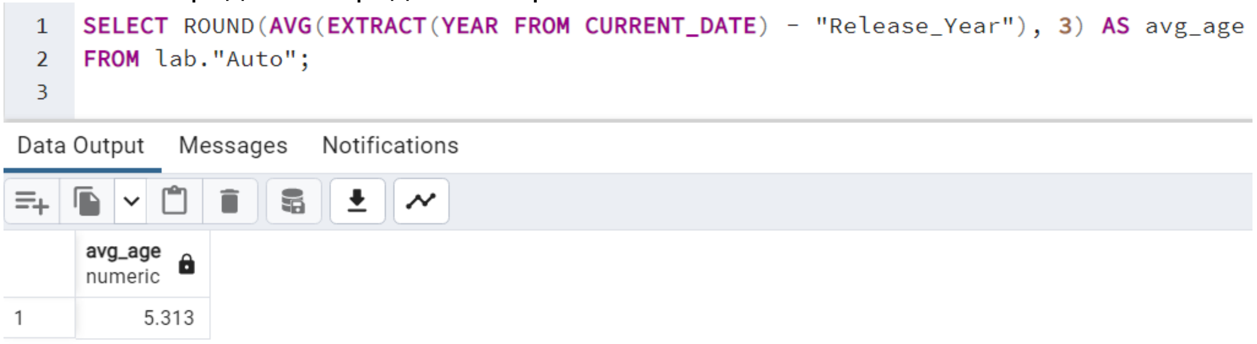
- Определить, каким количеством автомобилей каждой марки и модели владеет компания.



```sql
SELECT m."Name", COUNT(*)
FROM lab."Auto" a
JOIN lab."Model" m ON a."Mod_Code" = m."Mod_Code"
GROUP BY m."Name";
```

	Name character varying (40)	count bigint
1	Audi Q7	3
2	Porsche Panamera	3
3	Audi A6	3
4	Lexus LS	4
5	Jaguar F-Type	3
6	Mercedes-Benz E-Class	6
7	BMW X5	7
8	Tesla Model S	3

- Определить средний “возраст” автомобилей компании.

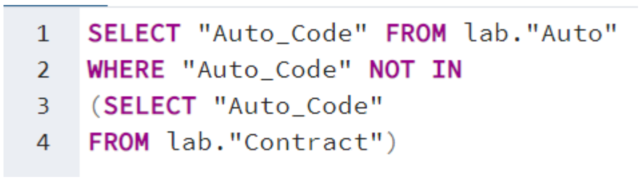


```sql
SELECT ROUND(AVG(EXTRACT(YEAR FROM CURRENT_DATE) - "Release_Year"), 3) AS avg_age
FROM lab."Auto";
```

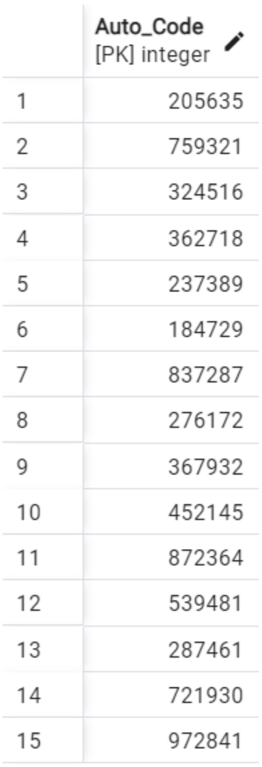
	avg_age numeric
1	5.313

Создание представлений:

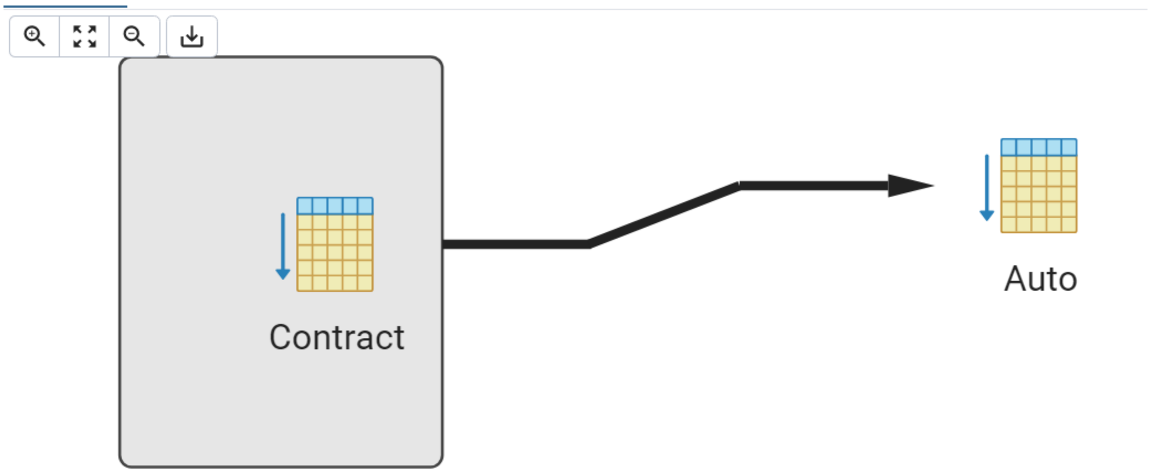
- Какой автомобиль ни разу не был в прокате?

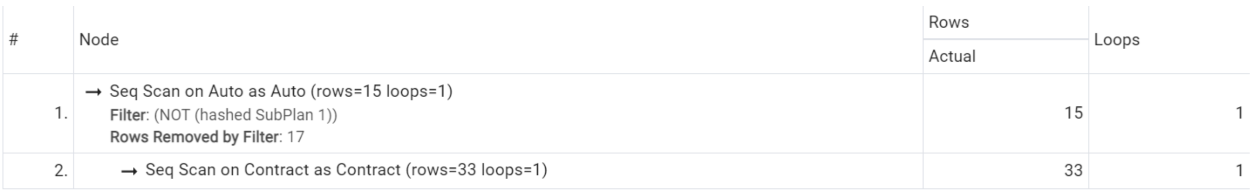


```sql
SELECT "Auto_Code" FROM lab."Auto"
WHERE "Auto_Code" NOT IN
(SELECT "Auto_Code"
FROM lab."Contract")
```



	Auto_Code [PK] integer
1	205635
2	759321
3	324516
4	362718
5	237389
6	184729
7	837287
8	276172
9	367932
10	452145
11	872364
12	539481
13	287461
14	721930
15	972841

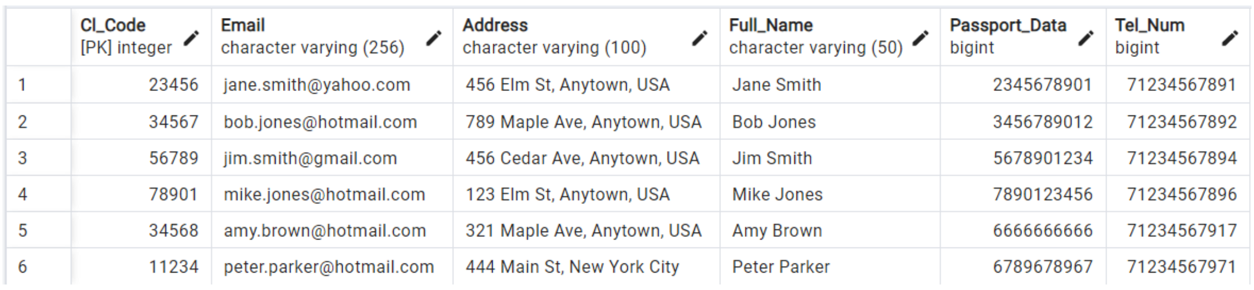




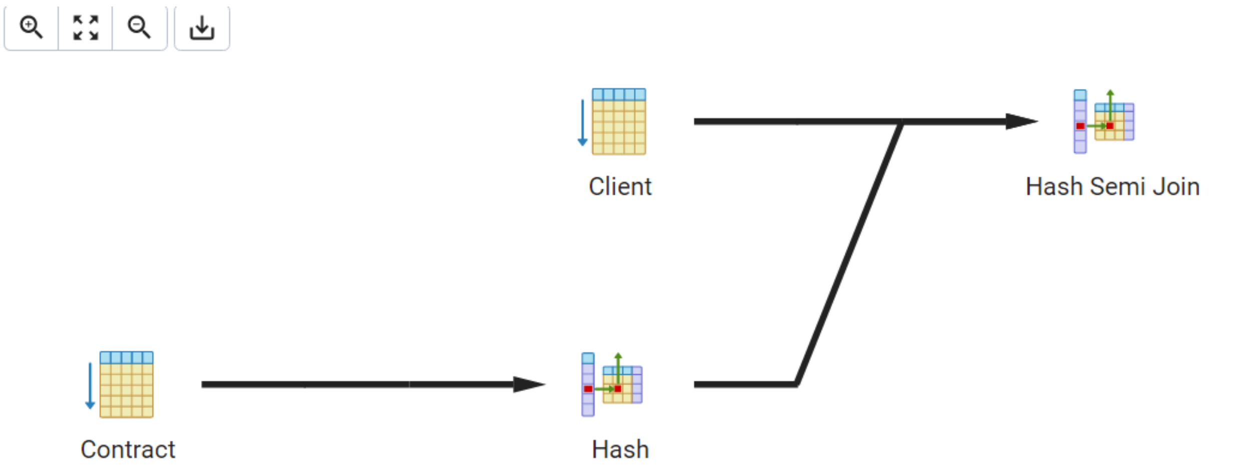
#	Node	Rows	Loops
		Actual	
1.	→ Seq Scan on Auto as Auto (rows=15 loops=1) Filter: (NOT (hashed SubPlan 1)) Rows Removed by Filter: 17	15	1
2.	→ Seq Scan on Contract as Contract (rows=33 loops=1)	33	1

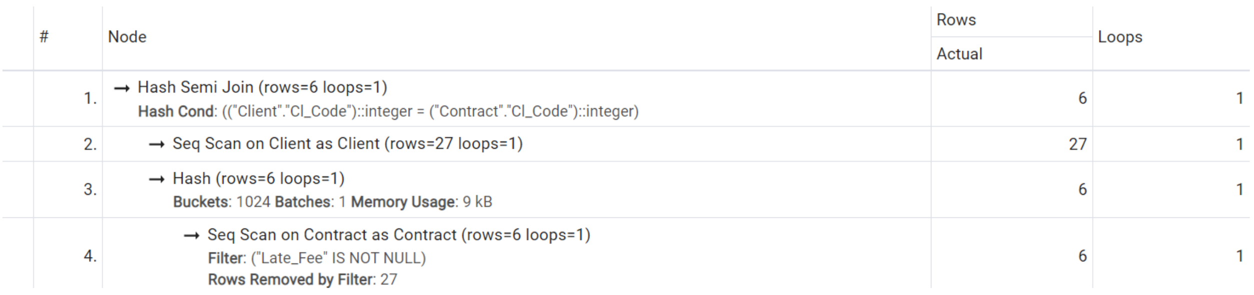
- Вывести данные клиентов, не вернувших автомобиль вовремя.

```sql
SELECT * FROM lab."Client"
WHERE "Cl_Code" IN
(SELECT "Cl_Code"
FROM lab."Contract"
WHERe "Late_Fee" IS NOT NULL);
```



	Cl_Code [PK] integer	Email character varying (256)	Address character varying (100)	Full_Name character varying (50)	Passport_Data bigint	Tel_Num bigint
1	23456	jane.smith@yahoo.com	456 Elm St, Anytown, USA	Jane Smith	2345678901	71234567891
2	34567	bob.jones@hotmail.com	789 Maple Ave, Anytown, USA	Bob Jones	3456789012	71234567892
3	56789	jim.smith@gmail.com	456 Cedar Ave, Anytown, USA	Jim Smith	5678901234	71234567894
4	78901	mike.jones@hotmail.com	123 Elm St, Anytown, USA	Mike Jones	7890123456	71234567896
5	34568	amy.brown@hotmail.com	321 Maple Ave, Anytown, USA	Amy Brown	6666666666	71234567917
6	11234	peter.parker@hotmail.com	444 Main St, New York City	Peter Parker	6789678967	71234567971

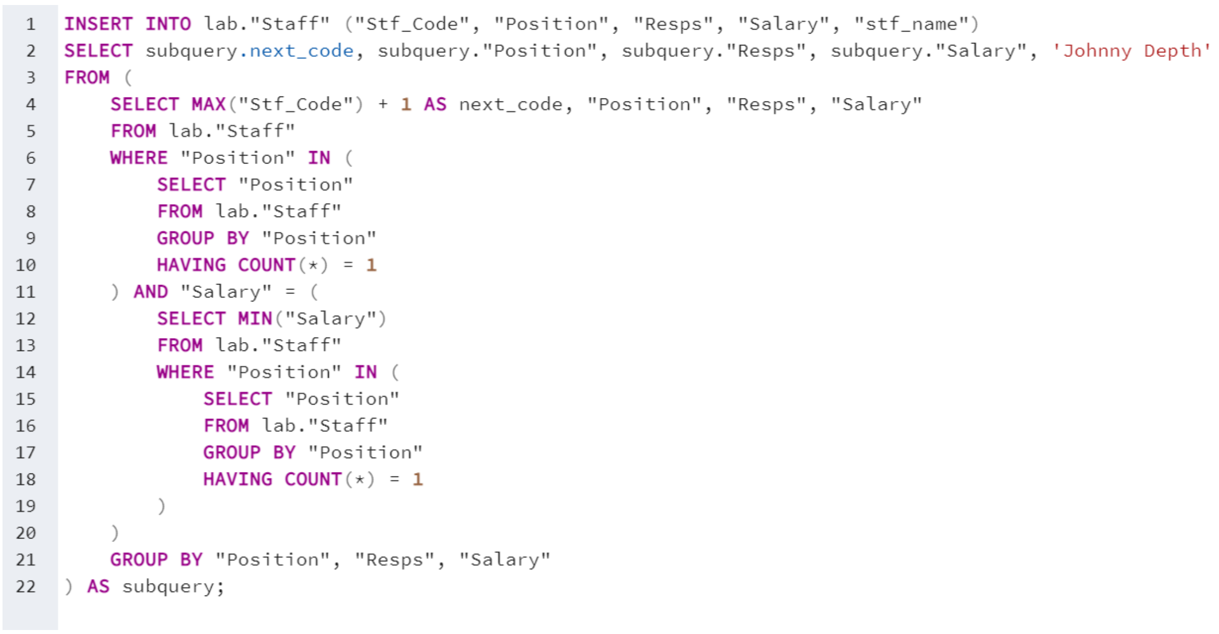




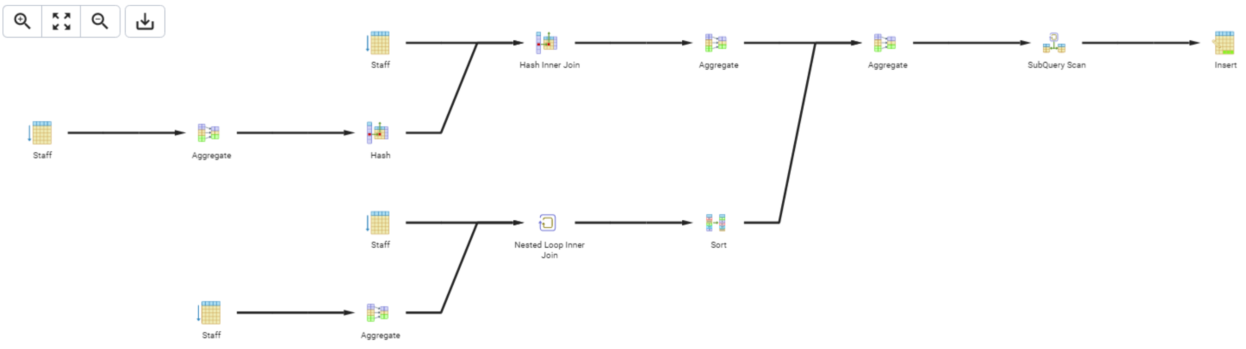
#	Node	Rows	Loops
		Actual	
1.	→ Hash Semi Join (rows=6 loops=1) **Hash Cond**: (("Client"."Cl_Code")::integer = ("Contract"."Cl_Code")::integer)	6	1
2.	→ Seq Scan on Client as Client (rows=27 loops=1)	27	1
3.	→ Hash (rows=6 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 9 kB	6	1
4.	→ Seq Scan on Contract as Contract (rows=6 loops=1) **Filter**: ("Late_Fee" IS NOT NULL) **Rows Removed by Filter**: 27	6	1

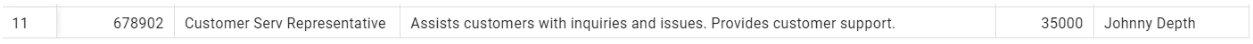
Запросы на модификацию данных

- **INSERT**



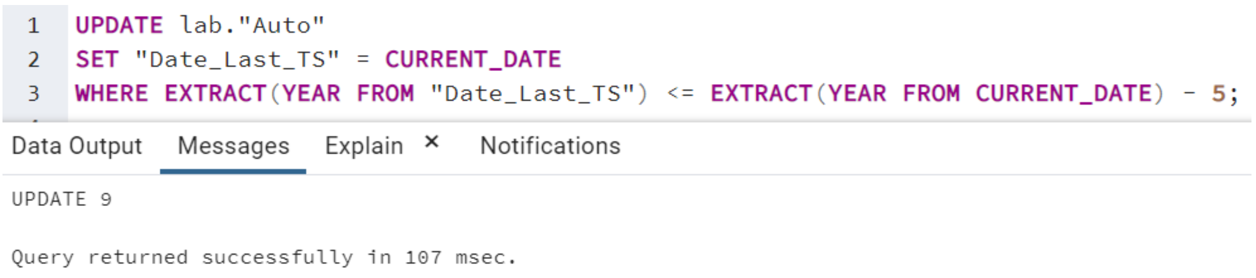
```sql
INSERT INTO lab."Staff" ("Stf_Code", "Position", "Resps", "Salary", "stf_name")
SELECT subquery.next_code, subquery."Position", subquery."Resps", subquery."Salary", 'Johnny Depth'
FROM (
    SELECT MAX("Stf_Code") + 1 AS next_code, "Position", "Resps", "Salary"
    FROM lab."Staff"
    WHERE "Position" IN (
        SELECT "Position"
        FROM lab."Staff"
        GROUP BY "Position"
        HAVING COUNT(*) = 1
    ) AND "Salary" = (
        SELECT MIN("Salary")
        FROM lab."Staff"
        WHERE "Position" IN (
            SELECT "Position"
            FROM lab."Staff"
            GROUP BY "Position"
            HAVING COUNT(*) = 1
        )
    )
    GROUP BY "Position", "Resps", "Salary"
) AS subquery;
```





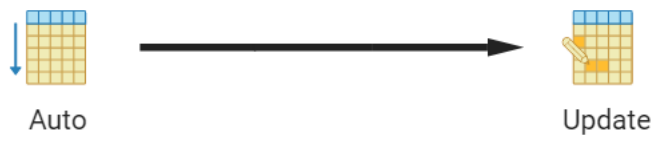
11	678902	Customer Serv Representative	Assists customers with inquiries and issues. Provides customer support.	35000	Johnny Depth

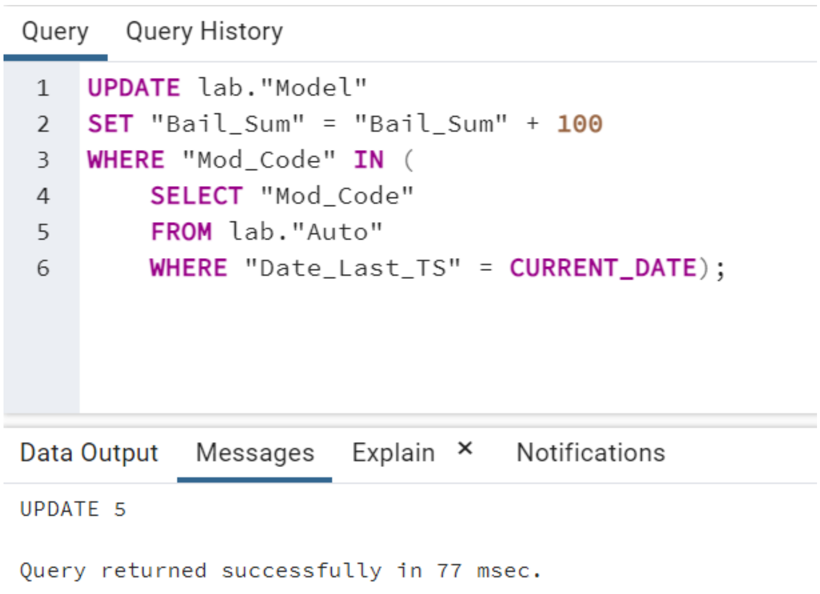
- **UPDATE**



```sql
UPDATE lab."Auto"
SET "Date_Last_TS" = CURRENT_DATE
WHERE EXTRACT(YEAR FROM "Date_Last_TS") <= EXTRACT(YEAR FROM CURRENT_DATE) - 5;
```

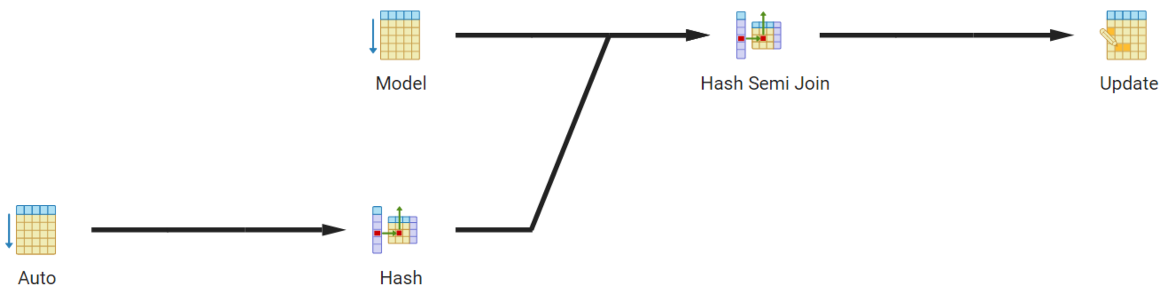
```
UPDATE 9

Query returned successfully in 107 msec.
```



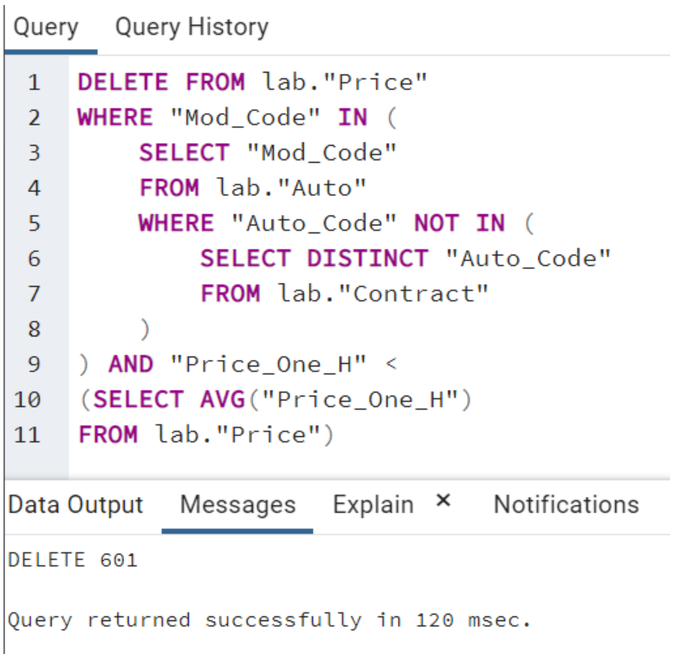


```sql
UPDATE lab."Model"
SET "Bail_Sum" = "Bail_Sum" + 100
WHERE "Mod_Code" IN (
    SELECT "Mod_Code"
    FROM lab."Auto"
    WHERE "Date_Last_TS" = CURRENT_DATE);
```

```
UPDATE 5

Query returned successfully in 77 msec.
```

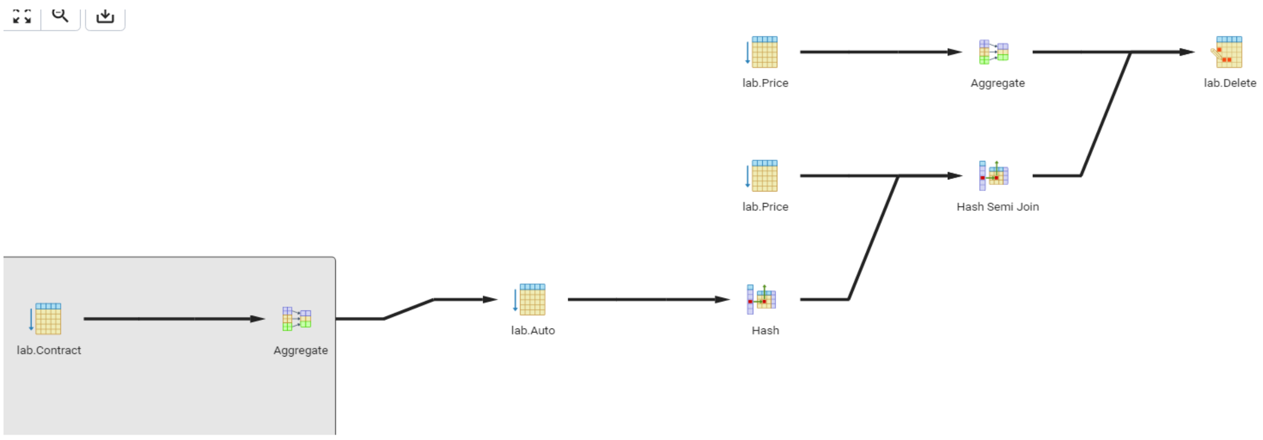


- **DELETE**



```
DELETE FROM lab."Price"
WHERE "Mod_Code" IN (
    SELECT "Mod_Code"
    FROM lab."Auto"
    WHERE "Auto_Code" NOT IN (
        SELECT DISTINCT "Auto_Code"
        FROM lab."Contract"
    )
) AND "Price_One_H" <
(SELECT AVG("Price_One_H")
FROM lab."Price")
```

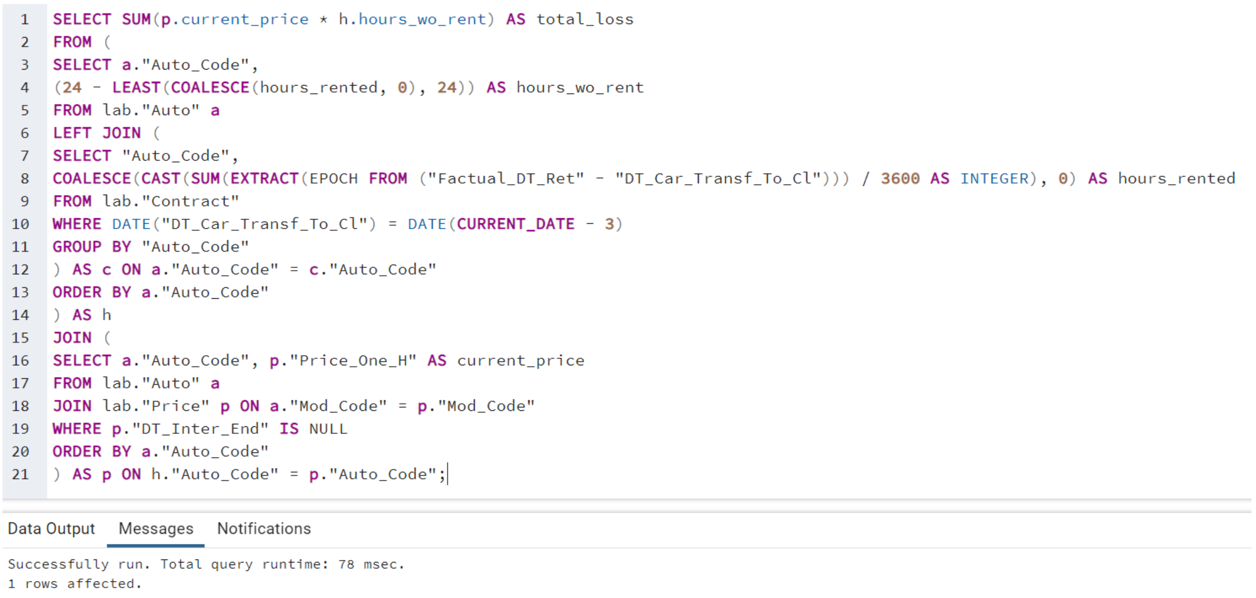
```
DELETE 601

Query returned successfully in 120 msec.
```



Создание индексов

Создадим простые индексы:

Используем код из предыдущего задания:

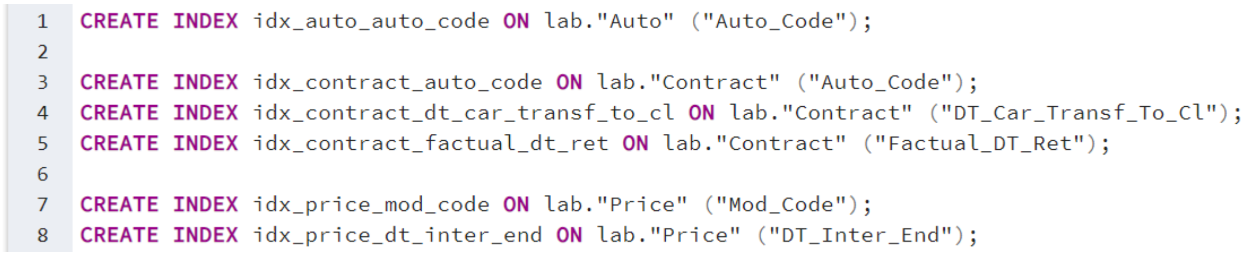


```
SELECT SUM(p.current_price * h.hours_wo_rent) AS total_loss
FROM (
SELECT a."Auto_Code",
(24 - LEAST(COALESCE(hours_rented, 0), 24)) AS hours_wo_rent
FROM lab."Auto" a
LEFT JOIN (
SELECT "Auto_Code",
COALESCE(CAST(SUM(EXTRACT(EPOCH FROM ("Factual_DT_Ret" - "DT_Car_Transf_To_Cl"))) / 3600 AS INTEGER), 0) AS hours_rented
FROM lab."Contract"
WHERE DATE("DT_Car_Transf_To_Cl") = DATE(CURRENT_DATE - 3)
GROUP BY "Auto_Code"
) AS c ON a."Auto_Code" = c."Auto_Code"
ORDER BY a."Auto_Code"
) AS h
JOIN (
SELECT a."Auto_Code", p."Price_One_H" AS current_price
FROM lab."Auto" a
JOIN lab."Price" p ON a."Mod_Code" = p."Mod_Code"
WHERE p."DT_Inter_End" IS NULL
ORDER BY a."Auto_Code"
) AS p ON h."Auto_Code" = p."Auto_Code";
```

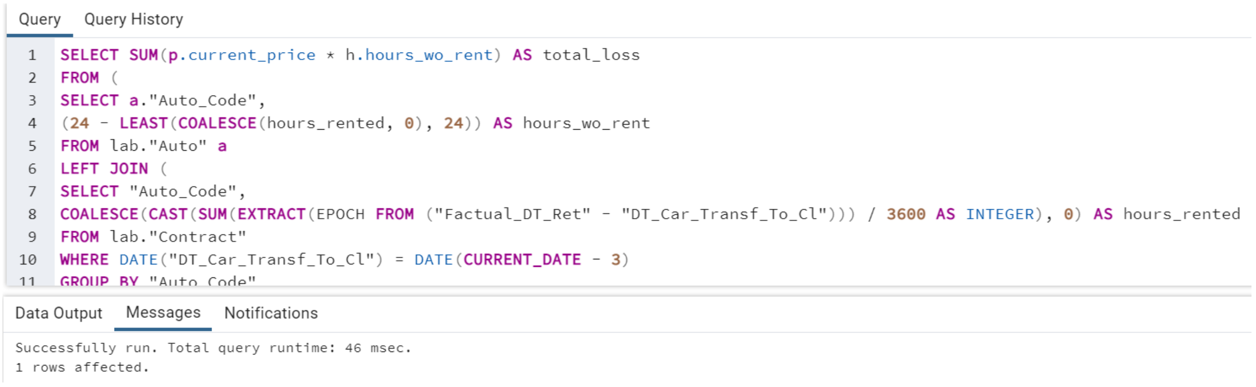
Data Output   Messages   Notifications

```
Successfully run. Total query runtime: 78 msec.
1 rows affected.
```

Без Индексации: время выполнения 78 мс



```
CREATE INDEX idx_auto_auto_code ON lab."Auto" ("Auto_Code");

CREATE INDEX idx_contract_auto_code ON lab."Contract" ("Auto_Code");
CREATE INDEX idx_contract_dt_car_transf_to_cl ON lab."Contract" ("DT_Car_Transf_To_Cl");
CREATE INDEX idx_contract_factual_dt_ret ON lab."Contract" ("Factual_DT_Ret");

CREATE INDEX idx_price_mod_code ON lab."Price" ("Mod_Code");
CREATE INDEX idx_price_dt_inter_end ON lab."Price" ("DT_Inter_End");
```



Query   Query History

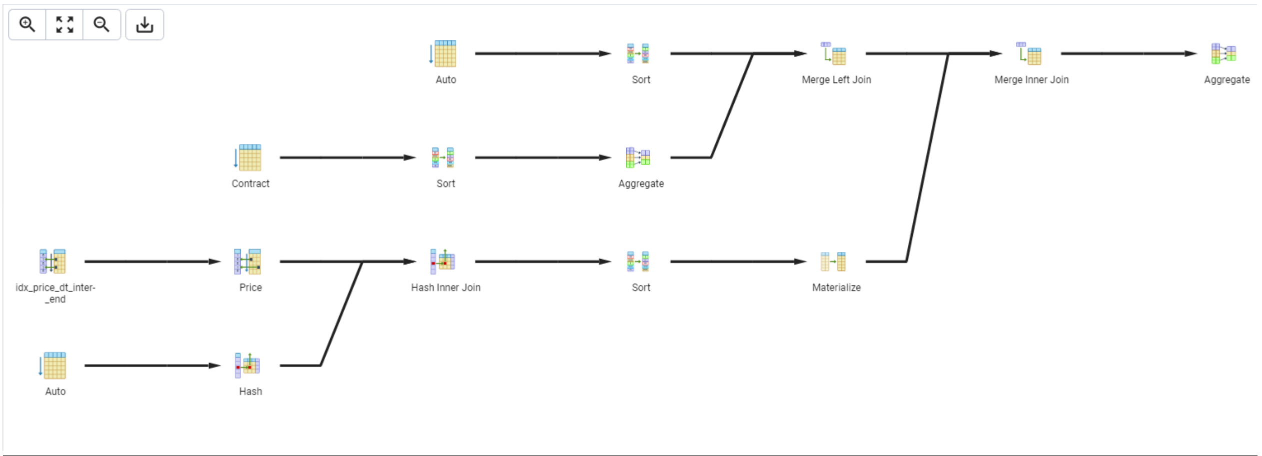
```
SELECT SUM(p.current_price * h.hours_wo_rent) AS total_loss
FROM (
SELECT a."Auto_Code",
(24 - LEAST(COALESCE(hours_rented, 0), 24)) AS hours_wo_rent
FROM lab."Auto" a
LEFT JOIN (
SELECT "Auto_Code",
COALESCE(CAST(SUM(EXTRACT(EPOCH FROM ("Factual_DT_Ret" - "DT_Car_Transf_To_Cl"))) / 3600 AS INTEGER), 0) AS hours_rented
FROM lab."Contract"
WHERE DATE("DT_Car_Transf_To_Cl") = DATE(CURRENT_DATE - 3)
GROUP BY "Auto_Code"
```

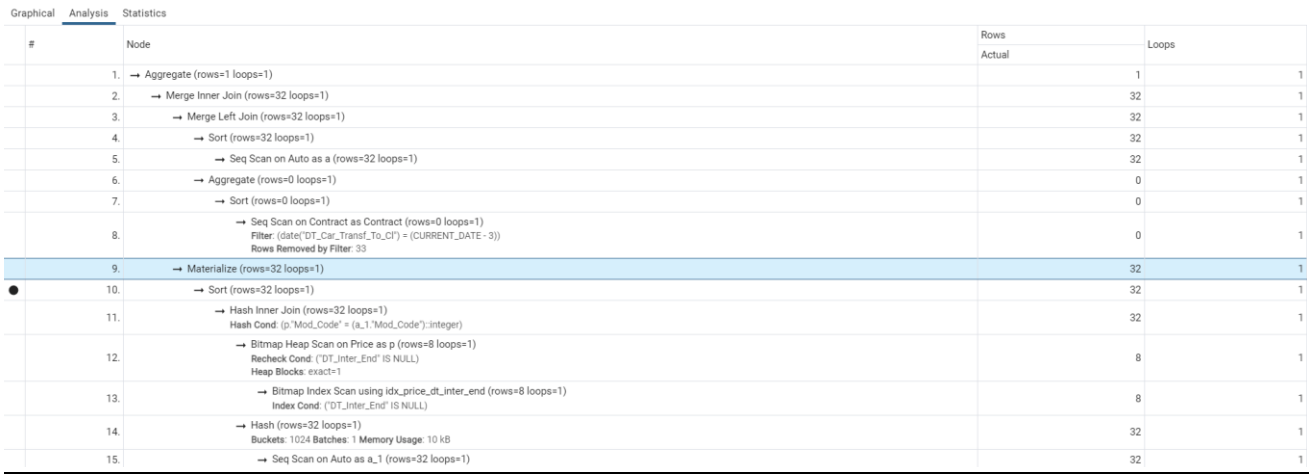
Data Output   Messages   Notifications

```
Successfully run. Total query runtime: 46 msec.
1 rows affected.
```

После индексации: время выполнения 46 мс

Итог: при помощи индексации мы ускорили время выполнения запроса на 32мс (~25%)





	#	Node	Rows	Loops
			Actual	
	1.	→ Aggregate (rows=1 loops=1)	1	1
	2.	→ Merge Inner Join (rows=32 loops=1)	32	1
	3.	→ Merge Left Join (rows=32 loops=1)	32	1
	4.	→ Sort (rows=32 loops=1)	32	1
	5.	→ Seq Scan on Auto as a (rows=32 loops=1)	32	1
	6.	→ Aggregate (rows=0 loops=1)	0	1
	7.	→ Sort (rows=0 loops=1)	0	1
	8.	→ Seq Scan on Contract as Contract (rows=0 loops=1) **Filter**: (date("DT_Car_Transf_To_Cl") = (CURRENT_DATE - 3)) **Rows Removed by Filter**: 33	0	1
	9.	→ Materialize (rows=32 loops=1)	32	1
●	10.	→ Sort (rows=32 loops=1)	32	1
	11.	→ Hash Inner Join (rows=32 loops=1) **Hash Cond**: (p."Mod_Code" = (a_1."Mod_Code")::integer)	32	1
	12.	→ Bitmap Heap Scan on Price as p (rows=8 loops=1) **Recheck Cond**: ("DT_Inter_End" IS NULL) **Heap Blocks**: exact=1	8	1
	13.	→ Bitmap Index Scan using idx_price_dt_inter_end (rows=8 loops=1) **Index Cond**: ("DT_Inter_End" IS NULL)	8	1
	14.	→ Hash (rows=32 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 10 kB	32	1
	15.	→ Seq Scan on Auto as a_1 (rows=32 loops=1)	32	1

Создадим составные индексы:

```sql
CREATE INDEX idx_time_in_rent
ON lab."Contract" ("Factual_DT_Ret", "DT_Car_Transf_To_Cl");
```

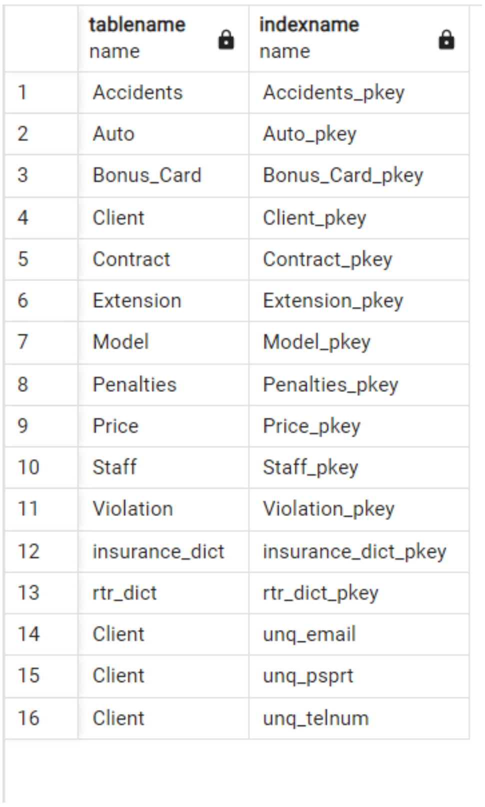
```
Successfully run. Total query runtime: 42 msec.
1 rows affected.
```

Итог: благодаря добавлению составного индекса удалось добиться ускорения выполнения запроса на 4 мс (1.8%)

Удалим Индексы:

17	Contract	idx_contract_auto_code
18	Contract	idx_contract_dt_car_transf_to_cl
19	Contract	idx_contract_factual_dt_ret
20	Price	idx_price_mod_code
21	Price	idx_price_dt_inter_end
22	Contract	idx_time_in_rent

```sql
DROP INDEX lab."idx_time_in_rent";
DROP INDEX lab."idx_auto_auto_code";
DROP INDEX lab."idx_contract_auto_code";
DROP INDEX lab."idx_contract_dt_car_transf_to_cl";
DROP INDEX lab."idx_contract_factual_dt_ret";
DROP INDEX lab."idx_price_mod_code";
DROP INDEX lab."idx_price_dt_inter_end";
DROP INDEX lab."idx_contract_auto_code";
DROP INDEX

Query returned successfully in 69 msec.
```



	tablename name	indexname name
1	Accidents	Accidents_pkey
2	Auto	Auto_pkey
3	Bonus_Card	Bonus_Card_pkey
4	Client	Client_pkey
5	Contract	Contract_pkey
6	Extension	Extension_pkey
7	Model	Model_pkey
8	Penalties	Penalties_pkey
9	Price	Price_pkey
10	Staff	Staff_pkey
11	Violation	Violation_pkey
12	insurance_dict	insurance_dict_pkey
13	rtr_dict	rtr_dict_pkey
14	Client	unq_email
15	Client	unq_psprt
16	Client	unq_telnum

Все индексы удалены.

Выводы:

Нами были выполнены SELECT запросы к созданной базе данных, я ознакомился с созданием запросов INSERT, UPDATE и DELETE, а также с графическим представлением запросов. Мы также изучили, как создавать простые и составные индексы, и показали, что это позволяет сокращать количество этапов выполнения запросов и снижать время выполнения.