

Практика по базам данных

ОТЧЁТ

Родионов Максим

Группа 23.Б15

Предметная область: «Продажа автомобилей»

Содержание

1	ОПИСАНИЕ СИСТЕМЫ	3
1.1	Требования	3
1.2	Модель данных	3
1.3	Функциональность	3
1.3.1	Серверная часть	3
1.3.2	Клиентская часть	5
2	СКРИПТЫ	8
2.1	Серверная часть	8
2.2	Клиентская часть	20
3	ПРИЛОЖЕНИЕ	26
3.1	Создание базы данных	26

1 ОПИСАНИЕ СИСТЕМЫ

1.1 Требования

Задача заключается в автоматизации работы компании, которая занимается продажей новых и подержанных автомобилей. В системе должен вестись учет продаж автомобилей, причем должны фиксироваться не только модели и года выпуска, но и другие базовые характеристики новых и подержанных автомобилей, а для последних – еще и показатели их состояния. При учете продаж следует автоматически отслеживать рейтинг сотрудников компании (в первую очередь - количество продаж), поскольку эти данные впоследствии будут передаваться в отдельную расчетную систему. Кроме того, система должна хранить информацию для создания статистических отчетов по клиентам (физическим и юридическим лицам).

1.2 Модель данных

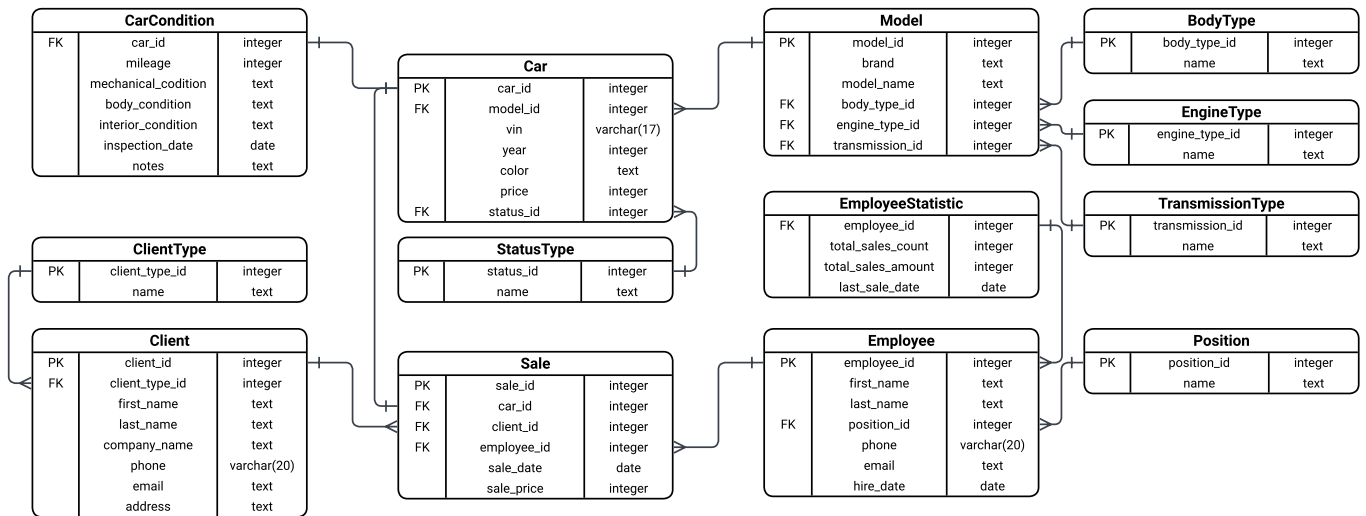


Рис. 1: ER-диаграмма

1.3 Функциональность

1.3.1 Серверная часть

Серверная логика реализована с использованием хранимых процедур, триггеров и представлений. Основные компоненты:

Хранимые процедуры / функции	Реализация	Комментарии
Регистрация нового клиента	—	
Оформление продажи	—	автоматически обновляет статус авто и статистику
Резервирование автомобиля	<code>reserve_car()</code>	меняет статус на «Reserved»
Возврат продажи	<code>refund_sale()</code>	удаляет продажу и восстанавливает статус
Обновление статистики сотрудника	<code>update_employee_stat()</code>	универсальная функция, которая вызывается в триггерах и процедурах
Отчёт по складу в JSON	<code>get_inventory_json()</code>	для интеграции с внешними системами
...		

Триггеры	Реализация	Комментарии
Автоматическое обновление статистики	<code>trg_update_stat_after_sale</code>	после каждой продажи
Смена статуса авто на «Sold»	<code>trg_set_car_sold_after_sale</code>	при оформлении продажи
Создание записи о состоянии авто	<code>trg_maintain_car_condition</code>	при добавлении авто
Проверка корректности VIN-кода	—	при добавлении авто
Запрет повторной продажи	<code>trg_prevent_sale_of_sold_car</code>	блокирует продажу уже проданного авто
Обновление даты последнего технического осмотра	—	
Проверка корректности пробега	—	не может уменьшаться при обновлении записи
...		

Представления	Реализация	Комментарии
Доступные автомобили	<code>v_available_cars</code>	для каталога

Представления	Реализация	Комментарии
История продаж	v_sales_report	с именами клиентов и сотрудников
Эффективность сотрудников	v_employee_performance	для рейтинга и отчётов
Статистика по моделям	—	для анализа прибыльности моделей
Автомобили, находящиеся в сервисе	—	
Статистика продаж по месяцам с динамикой и сравнением показателей	—	
История покупок клиентов	—	
...		

1.3.2 Клиентская часть

Клиентская часть использует представления, процедуры и прямые SQL-запросы для формирования экранов и отчётов:

Экранные формы основные	Дополнительные	Реализация (запрос)	Серверная часть
Реестр клиентов			
	Добавление клиента		
	Изменение данных клиента		
	Фильтр клиентов	(05) Клиенты, купившие автомобили марки Volkswagen (с именем/названием, email, адресом)	
Реестр автомобилей			

Экранные формы основные	Дополнительные	Реализация (запрос)	Серверная часть
	Добавление авто		<code>trg_maintain_car_condition</code> (для каскадного добавления CarCondition)
	Фильтры авто (по бренду, году, цене, статусу, ...)	(01) Автомобили без записей в Sale и (06) Доступные автомобили с пробегом более 50 000 км	<code>v_available_cars</code> (доступные авто)
	Изменение данных авто		
	Резервирование авто		<code>reserve_car()</code>
	Удаление авто		<code>trg_maintain_car_condition</code> (для каскадного удаления CarCondition)
	Рейтинг авто	(12) Автомобили, цена которых выше средней по их бренду	
Реестр сотрудников			
	Добавление сотрудника		
	Изменение данных сотрудника		
Реестр продаж			
	Добавление продажи		<code>trg_set_car_sold_after_sale</code> (автоматическая установка статуса «Продано» для авто) и <code>trg_update_stat_after_sale</code> (автоматическое обновление статистики продавца)
	Удаление продажи		<code>refund_sale()</code>

Экранные формы основные	Дополнительные	Реализация (запрос)	Серверная часть
Статистика по авто			
	Средняя цена	(02) Средняя стоимость автомобилей по каждому бренду	
	Рейтинг брендов	(07) Бренды с наибольшим числом продаж за последние 3 месяца	
Статистика по сотрудникам			
	Обновление статистики сотрудника		update_employee_stat()
	Рейтинг сотрудников	(11) Сотрудники с наибольшим числом продаж в текущем месяце	v_employee_performance
	Фильтр сотрудников	(08) Сотрудники, не совершавшие продажи последние 2 месяца	
Склад			
	Аналитика/отчёт		get_inventory_json()
Отчёты по продажам			
	История продаж		v_sales_report
	Аналитика по прибыли	(13) Средняя разница между первоначальной и продажной ценой по брендам	

Экранные формы основные	Дополнительные	Реализация (запрос)	Серверная часть
	Отчёт по месяцам	(04) Количество и выручка по месяцам за последний год	
	Топ продаж	(10) Три самых доро- гих продажи в исто- рии	

2 СКРИПТЫ

2.1 Серверная часть

Хранимые функции представлены на листинге 1.

```

1  -- =====
2  -- 1) Универсальная функция обновления статистики по одному сотруднику
3  -- =====
4  CREATE OR REPLACE FUNCTION update_employee_stat(p_employee_id INT)
5      RETURNS VOID LANGUAGE plpgsql AS $$ BEGIN
6      UPDATE EmployeeStatistic emp_stat
7      SET total_sales_count = sub.cnt,
8          total_sales_amount = sub.amt,
9          last_sale_date = sub.last_dt
10     FROM (
11         SELECT COUNT(Sale.sale_id) AS cnt,
12                COALESCE(SUM(Sale.sale_price), 0) AS amt,
13                MAX(Sale.sale_date) AS last_dt
14         FROM Sale
15         WHERE Sale.employee_id = p_employee_id
16     ) sub
17 WHERE emp_stat.employee_id = p_employee_id;
18
19 INSERT INTO EmployeeStatistic(
20     employee_id,
21     total_sales_count,
22     total_sales_amount,
23     last_sale_date
24 )

```



```

24 SELECT p_employee_id AS employee_id,
25        sub.cnt,
26        sub.amt,
27        sub.last_dt
28 FROM (
29         SELECT COUNT(sale_id) AS cnt,
30                COALESCE(SUM(sale_price), 0) AS amt,
31                MAX(sale_date) AS last_dt
32         FROM Sale
33         WHERE employee_id = p_employee_id
34        ) sub ON CONFLICT (employee_id) DO
35 UPDATE
36 SET total_sales_count = EXCLUDED.total_sales_count,
37    total_sales_amount = EXCLUDED.total_sales_amount,
38    last_sale_date = EXCLUDED.last_sale_date;
39
40 END;
41
42 $$;
43
44 -- CALL update_employee_stat(1);
45 --
46 -- =====
47 -- 2) Оценка стоимости склада автомобилей в формате JSON
48 -- =====
49 CREATE OR REPLACE FUNCTION get_inventory_json() RETURNS JSONB LANGUAGE
    plpgsql AS $$
50 DECLARE report_data JSONB;
51
52 tag_summary JSONB;
53
54 tag_financials JSONB;
55
56 tag_brands JSONB;
57
58 tag_status_breakdown JSONB;
59
60 tag_analytics JSONB;
61
62 tag_total_cars INT;

```

```

63
64 tag_sold_cars INT;
65
66 BEGIN
67 SELECT jsonb_build_object(
68     'total_cars',
69     COUNT(*),
70     'available_cars',
71     COUNT(*) FILTER (
72         WHERE st.name = 'Available'
73     ),
74     'reserved_cars',
75     COUNT(*) FILTER (
76         WHERE st.name = 'Reserved'
77     ),
78     'sold_cars',
79     COUNT(*) FILTER (
80         WHERE st.name = 'Sold'
81     )
82 ),
83 jsonb_build_object(
84     'total_inventory_value',
85     COALESCE(
86         SUM(Car.price) FILTER (
87             WHERE st.name = 'Available'
88         ),
89         0
90     ),
91     'total_reserved_value',
92     COALESCE(
93         SUM(Car.price) FILTER (
94             WHERE st.name = 'Reserved'
95         ),
96         0
97     ),
98     'total_sold_value',
99     COALESCE(
100        SUM(Car.price) FILTER (
101            WHERE st.name = 'Sold'
102        ),

```

```

103         0
104     ),
105     'avg_available_price',
106     COALESCE(
107         ROUND(
108             AVG(Car.price) FILTER (
109                 WHERE st.name = 'Available'
110             ),
111             2
112         ),
113         0
114     )
115 ),
116 COUNT(*),
117 COUNT(*) FILTER (
118     WHERE st.name = 'Sold'
119 ) INTO tag_summary,
120 tag_financials,
121 tag_total_cars,
122 tag_sold_cars
123 FROM Car
124 JOIN StatusType st ON Car.status_id = st.status_id;
125
126 SELECT jsonb_agg(
127     jsonb_build_object(
128         'brand',
129         brand,
130         'car_count',
131         car_count,
132         'total_value',
133         total_value,
134         'avg_price',
135         avg_price
136     )
137 ) INTO tag_brands
138 FROM (
139     SELECT Model.brand,
140         COUNT(*) AS car_count,
141         SUM(Car.price) AS total_value,
142         ROUND(AVG(Car.price), 2) AS avg_price

```

```

143         FROM Car
144             JOIN Model ON Car.model_id = Model.model_id
145             JOIN StatusType st ON Car.status_id = st.status_id
146         WHERE st.name = 'Available'
147         GROUP BY Model.brand
148     ) AS brand_data;
149
150 SELECT jsonb_agg(
151     jsonb_build_object(
152         'status',
153         status_name,
154         'count',
155         car_count,
156         'total_value',
157         total_value,
158         'min_price',
159         min_price,
160         'max_price',
161         max_price
162     )
163 ) INTO tag_status_breakdown
164 FROM (
165     SELECT st.name AS status_name,
166         COUNT(*) AS car_count,
167         COALESCE(SUM(Car.price), 0) AS total_value,
168         COALESCE(MIN(Car.price), 0) AS min_price,
169         COALESCE(MAX(Car.price), 0) AS max_price
170     FROM Car
171         JOIN StatusType st ON Car.status_id = st.status_id
172     GROUP BY st.name
173     ORDER BY st.name
174 ) AS status_data;
175
176 tag_analytics := jsonb_build_object(
177     'turnover_rate_percent',
178     CASE
179         WHEN tag_total_cars > 0 THEN ROUND(
180             (tag_sold_cars::NUMERIC / tag_total_cars) * 100,
181             2
182         )

```

```

183         ELSE 0
184     END
185 );
186
187 report_data := jsonb_build_object(
188     'generated_at',
189     CURRENT_TIMESTAMP,
190     'summary',
191     tag_summary,
192     'financials',
193     tag_financials,
194     'brands',
195     COALESCE(tag_brands, '[]'::JSONB),
196     'status_breakdown',
197     COALESCE(tag_status_breakdown, '[]'::JSONB),
198     'analytics',
199     tag_analytics
200 );
201
202 RETURN report_data;
203
204 END;
205
206 $$;
207
208 -- SELECT get_inventory_json();

```

Листинг 1: Хранимые функции

Хранимые процедуры представлены на листинге 2.

```

1  -- =====
2  -- 1) Бронирование автомобиля
3  -- =====
4  CREATE OR REPLACE PROCEDURE reserve_car(car_id_param INT) LANGUAGE
    plpgsql AS $$
5  DECLARE reserved_status_id INT;
6
7  current_status TEXT;
8
9  BEGIN
10 SELECT StatusType.name INTO current_status

```

```

11 FROM Car
12     JOIN StatusType ON Car.status_id = StatusType.status_id
13 WHERE Car.car_id = car_id_param;
14
15 IF current_status = 'Sold' THEN RAISE EXCEPTION 'Cannot reserve sold car'
16     ;
17 END IF;
18
19 SELECT status_id INTO reserved_status_id
20 FROM StatusType
21 WHERE name = 'Reserved'
22 LIMIT 1;
23
24 UPDATE Car
25 SET status_id = reserved_status_id
26 WHERE car_id = car_id_param;
27
28 END;
29
30 $$;
31
32 -- CALL reserve_car(3);
33 --
34 -- =====
35 -- 2) Возврат/отмена последней продажи автомобиля
36 -- =====
37 CREATE OR REPLACE PROCEDURE refund_sale(
38     IN p_car_id INT,
39     OUT refunded_sale_id INT,
40     OUT refunded_amount INT,
41     OUT refunded_car_id INT
42 ) LANGUAGE plpgsql AS $$
43 DECLARE v_sale RECORD;
44
45 BEGIN
46 SELECT * INTO v_sale
47 FROM Sale
48 WHERE car_id = p_car_id
49 ORDER BY sale_date DESC,

```

```

50     sale_id DESC
51 LIMIT 1;
52
53 IF NOT FOUND THEN RAISE EXCEPTION 'No sale found for car %',
54 p_car_id;
55
56 END IF;
57
58 refunded_sale_id := v_sale.sale_id;
59
60 refunded_amount := v_sale.sale_price;
61
62 refunded_car_id := v_sale.car_id;
63
64 DELETE FROM Sale
65 WHERE sale_id = v_sale.sale_id;
66
67 PERFORM update_employee_stat(v_sale.employee_id);
68
69 UPDATE Car
70 SET status_id = (
71     SELECT status_id
72     FROM StatusType
73     WHERE name = 'Available'
74     LIMIT 1
75 )
76 WHERE car_id = p_car_id;
77
78 END;
79
80 $$;
81
82 -- CALL refund_sale(1, NULL, NULL, NULL);

```

Листинг 2: Хранимые процедуры

Триггеры представлены на листинге 3.

```

1  -- =====
2  -- 1) Автоматически обновлять EmployeeStatistics при продаже
3  -- =====
4  CREATE OR REPLACE FUNCTION trg_update_stat_after_sale() RETURNS TRIGGER

```

```

    LANGUAGE plpgsql AS $$ BEGIN PERFORM update_employee_stat(NEW.
    employee_id);
5
6 RETURN NEW;
7
8 END;
9
10 $$;
11
12 CREATE TRIGGER trg_update_stat_after_sale
13 AFTER
14 INSERT ON Sale FOR EACH ROW EXECUTE FUNCTION trg_update_stat_after_sale()
    ;
15
16 -- =====
17 -- 2) Автоматически устанавливать статус "Sold" при продаже автомобиля
18 -- =====
19 CREATE OR REPLACE FUNCTION set_car_status_to_sold() RETURNS TRIGGER AS $$
    BEGIN
20 UPDATE Car
21 SET status_id = (
22     SELECT status_id
23     FROM StatusType
24     WHERE name = 'Sold'
25     LIMIT 1
26 )
27 WHERE car_id = NEW.car_id;
28
29 RETURN NEW;
30
31 END;
32
33 $$ LANGUAGE plpgsql;
34
35 CREATE TRIGGER trg_set_car_sold_after_sale
36 AFTER
37 INSERT ON Sale FOR EACH ROW EXECUTE FUNCTION set_car_status_to_sold();
38
39 -- =====
40 -- 3) Ведение данных CarCondition при добавлении/удалении автомобиля

```



```

41  -- =====
42  CREATE OR REPLACE FUNCTION maintain_car_condition() RETURNS TRIGGER AS $$
43      BEGIN IF (TG_OP = 'INSERT') THEN
44          INSERT INTO CarCondition (
45              car_id,
46              mileage,
47              mechanical_condition,
48              body_condition,
49              interior_condition,
50              inspection_date,
51              notes
52          )
53          VALUES (
54              NEW.car_id,
55              NULL,
56              NULL,
57              NULL,
58              NULL,
59              NULL,
60              NULL
61          );
62  RETURN NEW;
63
64  ELSIF (TG_OP = 'DELETE') THEN
65      DELETE FROM CarCondition
66      WHERE car_id = OLD.car_id;
67
68  RETURN OLD;
69
70  END IF;
71
72  RETURN NULL;
73
74  END;
75
76  $$ LANGUAGE plpgsql;
77
78  CREATE TRIGGER trg_maintain_car_condition
79  AFTER

```

```

80 INSERT
81     OR DELETE ON Car FOR EACH ROW EXECUTE FUNCTION maintain_car_condition
82     ();
83 -- =====
84 -- 4) Запрет на продажу автомобиля, если его статус - продан
85 -- =====
86 CREATE OR REPLACE FUNCTION prevent_sale_of_sold_car() RETURNS TRIGGER AS
87     $$
88 DECLARE current_status_name TEXT;
89 BEGIN
90 SELECT st.name INTO current_status_name
91 FROM Car
92     JOIN StatusType st ON Car.status_id = st.status_id
93 WHERE Car.car_id = NEW.car_id;
94
95 IF current_status_name = 'Sold' THEN RAISE EXCEPTION 'Cannot sell car %:
96     it is already sold',
97     NEW.car_id;
98 END IF;
99
100 RETURN NEW;
101
102 END;
103
104 $$ LANGUAGE plpgsql;
105
106 CREATE TRIGGER trg_prevent_sale_of_sold_car BEFORE
107 INSERT ON Sale FOR EACH ROW EXECUTE FUNCTION prevent_sale_of_sold_car();

```

Листинг 3: Триггеры

Представления представлены на листинге 4.

```

1 -- =====
2 -- 1) Доступные автомобили с подробной информацией о модели
3 -- =====
4 CREATE OR REPLACE VIEW v_available_cars AS
5 SELECT Car.car_id,
6     Model.brand,

```

```

7      Model.model_name ,
8      Car.year ,
9      CarCondition.mileage ,
10     Car.color ,
11     Car.price ,
12     BodyType.name AS body_type ,
13     EngineType.name AS engine_type_name
14 FROM Car
15     JOIN Model USING (model_id)
16     JOIN BodyType ON Model.body_type_id = BodyType.body_type_id
17     JOIN EngineType ON Model.engine_type_id = EngineType.engine_type_id
18     JOIN StatusType ON Car.status_id = StatusType.status_id
19     JOIN CarCondition ON CarCondition.car_id = Car.car_id
20 WHERE StatusType.name = 'Available';
21
22 SELECT *
23 FROM v_available_cars;
24
25 -- =====
26 -- 2) Полная история продаж с именами
27 -- =====
28 CREATE OR REPLACE VIEW v_sales_report AS
29 SELECT Sale.sale_id AS sale_id ,
30        Sale.sale_date AS sale_date ,
31        Sale.sale_price AS sale_amount ,
32        Model.brand || ' ' || Model.model_name AS automobile ,
33        CASE
34            WHEN Client.client_type_id = 1 THEN Client.first_name || ' ' ||
Client.last_name
35            ELSE Client.company_name
36        END AS customer ,
37        Employee.first_name || ' ' || Employee.last_name AS salesperson
38 FROM Sale
39     JOIN Employee ON Sale.employee_id = Employee.employee_id
40     JOIN Client ON Sale.client_id = Client.client_id
41     JOIN Car ON Sale.car_id = Car.car_id
42     JOIN Model ON Car.model_id = Model.model_id;
43
44 SELECT *
45 FROM v_sales_report;

```

```

46
47 -- =====
48 -- 3) Статистика сотрудников с указанием имени
49 -- =====
50 CREATE OR REPLACE VIEW v_employee_performance AS
51 SELECT Employee.employee_id,
52        Employee.first_name,
53        Employee.last_name,
54        Position.name AS position,
55        emp_stat.total_sales_count,
56        emp_stat.total_sales_amount,
57        emp_stat.last_sale_date
58 FROM Employee
59        JOIN Position ON Employee.position_id = Position.position_id
60        JOIN EmployeeStatistic emp_stat ON Employee.employee_id = emp_stat.
        employee_id;
61
62 SELECT *
63 FROM v_employee_performance;

```

Листинг 4: Представления

2.2 Клиентская часть

Запросы для экранных форм и отчетов представлены на листинге 5.

```

1 -- =====
2 -- 1) Автомобили, которые никогда не были проданы
3 -- =====
4 SELECT Car.car_id,
5        Model.brand || ' ' || Model.model_name AS model,
6        Car.year,
7        Car.price,
8        CarCondition.inspection_date
9 FROM Car
10        JOIN Model ON Car.model_id = Model.model_id
11        LEFT JOIN Sale ON Car.car_id = Sale.car_id
12        JOIN CarCondition ON CarCondition.car_id = Car.car_id
13 WHERE Sale.sale_id IS NULL
14 ORDER BY Car.year DESC;
15
16 -- =====

```

```

17 -- 2) Средняя цена по бренду
18 -- =====
19 SELECT Model.brand,
20        ROUND(AVG(Car.price), 2) AS average_price
21 FROM Car
22        JOIN Model ON Car.model_id = Model.model_id
23 GROUP BY Model.brand
24 ORDER BY average_price DESC;
25
26 -- =====
27 -- 3) Средний пробег по типу кузова для автомобилей
28 -- =====
29 SELECT bt.name AS body_type,
30        COUNT(*) AS cars_count,
31        ROUND(AVG(cc.mileage), 0) AS avg_mileage
32 FROM Car
33        JOIN Model ON Car.model_id = Model.model_id
34        JOIN BodyType bt ON Model.body_type_id = bt.body_type_id
35        JOIN CarCondition cc ON Car.car_id = cc.car_id
36 WHERE cc.mileage > 0
37 GROUP BY bt.name
38 HAVING COUNT(*) >= 1
39 ORDER BY avg_mileage DESC;
40
41 -- =====
42 -- 4) Ежемесячный отчет о продажах за прошлый год
43 -- =====
44 SELECT EXTRACT(
45        YEAR
46        FROM sale_date
47    ) AS year,
48    EXTRACT(
49        MONTH
50        FROM sale_date
51    ) AS MONTH,
52    TO_CHAR(sale_date, 'Month') AS month_name,
53    COUNT(*) AS sales_count,
54    SUM(sale_amount) AS total_revenue
55 FROM v_sales_report
56 WHERE sale_date >= CURRENT_DATE - INTERVAL '1 year'

```

```

57 GROUP BY year,
58     MONTH,
59     month_name
60 ORDER BY year,
61     MONTH;
62
63 -- =====
64 -- 5) Клиенты, купившие Volkswagen
65 -- =====
66 SELECT Client.client_id,
67     CASE
68         WHEN Client.client_type_id = 1 THEN Client.first_name || ' ' ||
        Client.last_name
69         ELSE Client.company_name
70     END AS customer,
71     Client.email,
72     Client.address,
73     Model.brand || ' ' || Model.model_name AS automobile
74 FROM Client
75     JOIN Sale ON Client.client_id = Sale.client_id
76     JOIN Car ON Sale.car_id = Car.car_id
77     JOIN Model ON Car.model_id = Model.model_id
78 WHERE Model.brand = 'Volkswagen';
79
80 -- =====
81 -- 6) В наличии автомобили с пробегом более 50 000 км
82 -- =====
83 SELECT car_id,
84     brand,
85     model_name,
86     year,
87     mileage,
88     color,
89     price
90 FROM v_available_cars
91 WHERE mileage > 50000
92 ORDER BY mileage DESC;
93
94 -- =====
95 -- 7) Бренды с самыми высокими продажами за последние 3 месяца

```

```

96  -- =====
97  SELECT SPLIT_PART(automobile, ' ', 1) AS brand,
98         COUNT(*) AS sales_count,
99         SUM(sale_amount) AS total_revenue
100 FROM v_sales_report
101 WHERE sale_date >= CURRENT_DATE - INTERVAL '3 months'
102 GROUP BY SPLIT_PART(automobile, ' ', 1)
103 HAVING COUNT(*) >= 1
104 ORDER BY sales_count DESC,
105         total_revenue DESC;
106
107  -- =====
108  -- 8) Сотрудники, у которых не было продаж за последние 2 месяца
109  -- =====
110  SELECT e.employee_id,
111         e.first_name || ' ' || e.last_name AS full_name,
112         p.name AS position
113  FROM Employee e
114       JOIN Position p ON e.position_id = p.position_id
115  WHERE NOT EXISTS (
116         SELECT 1
117         FROM Sale s
118         WHERE s.employee_id = e.employee_id
119               AND s.sale_date >= CURRENT_DATE - INTERVAL '2 months'
120       )
121  ORDER BY e.hire_date DESC;
122
123  -- =====
124  -- 9) Все контактные адреса электронной почты (клиенты + сотрудники)
125  -- =====
126  SELECT email
127  FROM Client
128  WHERE email IS NOT NULL
129  UNION
130  SELECT email
131  FROM Employee
132  WHERE email IS NOT NULL;
133
134  -- =====
135  -- 10) Топ-3 самых дорогих продаж

```

```

136 -- =====
137 SELECT *
138 FROM Sale
139 WHERE sale_price IN (
140     SELECT DISTINCT sale_price
141     FROM Sale
142     ORDER BY sale_price DESC
143     LIMIT 3
144 )
145 ORDER BY sale_price DESC;
146
147 -- =====
148 -- 11) Самые прибыльные сотрудники за текущий месяц
149 -- =====
150 SELECT es.employee_id,
151     Employee.first_name || ' ' || Employee.last_name AS salesperson,
152     es.total_sales_amount,
153     (
154         SELECT COUNT(*)
155         FROM Sale
156         WHERE employee_id = es.employee_id
157             AND EXTRACT(
158                 MONTH
159                 FROM sale_date
160             ) = EXTRACT(
161                 MONTH
162                 FROM CURRENT_DATE
163             )
164             AND EXTRACT(
165                 YEAR
166                 FROM sale_date
167             ) = EXTRACT(
168                 YEAR
169                 FROM CURRENT_DATE
170             )
171     ) AS sales
172 FROM Employee
173     JOIN EmployeeStatistic es ON Employee.employee_id = es.employee_id
174 WHERE es.total_sales_amount > 0
175 ORDER BY sales DESC;

```



```

176
177 -- =====
178 -- 12) Автомобили по цене выше средней для своей марки
179 -- =====
180 SELECT ac.car_id,
181        ac.brand,
182        ac.model_name,
183        ac.price,
184        (
185            SELECT ROUND(AVG(price), 2)
186            FROM Car c2
187            JOIN Model m2 ON c2.model_id = m2.model_id
188            WHERE m2.brand = ac.brand
189            AND c2.status_id = (
190                SELECT status_id
191                FROM StatusType
192                WHERE name = 'Available'
193            )
194        ) AS avg_brand_price
195 FROM v_available_cars ac
196 WHERE ac.price > (
197     SELECT AVG(c2.price)
198     FROM Car c2
199     JOIN Model m2 ON c2.model_id = m2.model_id
200     WHERE m2.brand = ac.brand
201     AND c2.status_id = (
202         SELECT status_id
203         FROM StatusType
204         WHERE name = 'Available'
205     )
206 );
207
208 -- =====
209 -- 13) Средняя разница в цене между изначальноными и проданными автомобилям
210 -- и по маркам
211 -- =====
212 SELECT Model.brand,
213        ROUND(AVG(Sale.sale_price - Car.price), 2) AS avg_price_difference,
214        COUNT(*) AS sales_count,
215        ROUND(AVG(Car.price), 2) AS avg_original_price,

```

```

215     ROUND(AVG(Sale.sale_price), 2) AS avg_sold_price
216 FROM Sale
217     JOIN Car ON Sale.car_id = Car.car_id
218     JOIN Model ON Car.model_id = Model.model_id
219 GROUP BY Model.brand
220 ORDER BY avg_price_difference DESC;

```

Листинг 5: Запросы

3 ПРИЛОЖЕНИЕ

3.1 Создание базы данных

Создание таблиц и индексов представлено на листинге 6.

```

1  -- =====
2  -- Справочные таблицы (словари)
3  -- =====
4  CREATE TABLE ClientType (
5      client_type_id SERIAL PRIMARY KEY,
6      name TEXT NOT NULL UNIQUE
7  );
8
9  CREATE TABLE BodyType (
10     body_type_id SERIAL PRIMARY KEY,
11     name TEXT NOT NULL UNIQUE
12 );
13
14 CREATE TABLE EngineType (
15     engine_type_id SERIAL PRIMARY KEY,
16     name TEXT NOT NULL UNIQUE
17 );
18
19 CREATE TABLE TransmissionType (
20     transmission_id SERIAL PRIMARY KEY,
21     name TEXT NOT NULL UNIQUE
22 );
23
24 CREATE TABLE Position (
25     position_id SERIAL PRIMARY KEY,
26     name TEXT NOT NULL UNIQUE

```

```

27 );
28
29 CREATE TABLE StatusType (
30     status_id SERIAL PRIMARY KEY,
31     name TEXT NOT NULL UNIQUE
32 );
33
34 -- =====
35 -- Основные таблицы
36 -- =====
37 CREATE TABLE Client (
38     client_id SERIAL PRIMARY KEY,
39     client_type_id INT REFERENCES ClientType(client_type_id),
40     first_name TEXT,
41     last_name TEXT,
42     company_name TEXT,
43     phone VARCHAR(20),
44     email TEXT,
45     address TEXT
46 );
47
48 CREATE TABLE Employee (
49     employee_id SERIAL PRIMARY KEY,
50     first_name TEXT NOT NULL,
51     last_name TEXT NOT NULL,
52     position_id INT REFERENCES Position(position_id),
53     phone VARCHAR(20),
54     email TEXT,
55     hire_date DATE NOT NULL
56 );
57
58 CREATE TABLE Model (
59     model_id SERIAL PRIMARY KEY,
60     brand TEXT NOT NULL,
61     model_name TEXT NOT NULL,
62     body_type_id INT REFERENCES BodyType(body_type_id),
63     engine_type_id INT REFERENCES EngineType(engine_type_id),
64     transmission_id INT REFERENCES TransmissionType(transmission_id)
65 );
66

```

```

67 CREATE TABLE Car (
68     car_id SERIAL PRIMARY KEY,
69     model_id INT NOT NULL REFERENCES Model(model_id),
70     vin VARCHAR(17) NOT NULL UNIQUE,
71     year INT CHECK (
72         year >= 1900
73         AND year <= EXTRACT(
74             YEAR
75             FROM CURRENT_DATE
76         ) + 1
77     ),
78     color TEXT,
79     price INT CHECK (price >= 0),
80     status_id INT REFERENCES StatusType(status_id)
81 );
82
83 CREATE TABLE CarCondition (
84     car_id INT PRIMARY KEY REFERENCES Car(car_id) ON DELETE CASCADE,
85     mileage INT CHECK (mileage >= 0),
86     mechanical_condition TEXT,
87     body_condition TEXT,
88     interior_condition TEXT,
89     inspection_date DATE,
90     notes TEXT
91 );
92
93 CREATE TABLE Sale (
94     sale_id SERIAL PRIMARY KEY,
95     car_id INT NOT NULL REFERENCES Car(car_id),
96     client_id INT NOT NULL REFERENCES Client(client_id),
97     employee_id INT NOT NULL REFERENCES Employee(employee_id),
98     sale_date DATE NOT NULL,
99     sale_price INT CHECK (sale_price >= 0)
100 );
101
102 CREATE TABLE EmployeeStatistic (
103     employee_id INT PRIMARY KEY REFERENCES Employee(employee_id) ON
104     DELETE CASCADE,
105     total_sales_count INT DEFAULT 0 CHECK (total_sales_count >= 0),
106     total_sales_amount INT DEFAULT 0 CHECK (total_sales_amount >= 0),

```

```

106     last_sale_date DATE
107 );
108
109 -- =====
110 -- Индексы
111 -- =====
112 CREATE INDEX idx_car_model ON Car(model_id);
113
114 CREATE INDEX idx_sale_employee ON Sale(employee_id);
115
116 CREATE INDEX idx_sale_client ON Sale(client_id);
117
118 CREATE INDEX idx_sale_date ON Sale(sale_date);
119
120 CREATE INDEX idx_car_status ON Car(status_id);
121
122 CREATE INDEX idx_model_brand_model ON Model(brand, model_name);
123
124 CREATE INDEX idx_car_year ON Car(year);

```

Листинг 6: Создание таблиц и индексов

Заполнение таблиц тестовыми данными представлено на листинге 7.

```

1 -- =====
2 -- Значения для справочных словарей
3 -- =====
4 INSERT INTO ClientType (name)
5 VALUES ('Individual'),
6         ('Legal') ON CONFLICT DO NOTHING;
7
8 INSERT INTO StatusType (name)
9 VALUES ('Available'),
10         ('Reserved'),
11         ('Sold'),
12         ('In Service') ON CONFLICT DO NOTHING;
13
14 INSERT INTO Position (name)
15 VALUES ('Salesperson'),
16         ('Manager'),
17         ('Director') ON CONFLICT DO NOTHING;
18

```

```

19 INSERT INTO BodyType (name)
20 VALUES ('Sedan'),
21         ('Hatchback'),
22         ('SUV'),
23         ('CUV'),
24         ('Coupe'),
25         ('Wagon') ON CONFLICT DO NOTHING;
26
27 INSERT INTO EngineType (name)
28 VALUES ('Petrol'),
29         ('Diesel'),
30         ('Hybrid'),
31         ('Electric') ON CONFLICT DO NOTHING;
32
33 INSERT INTO TransmissionType (name)
34 VALUES ('Manual'),
35         ('Automatic') ON CONFLICT DO NOTHING;
36
37 -- =====
38 -- Сотрудники
39 -- =====
40 INSERT INTO Employee (
41     first_name,
42     last_name,
43     position_id,
44     phone,
45     email,
46     hire_date
47 )
48 VALUES (
49     'Ivan',
50     'Petrov',
51     1,
52     '+7-900-111-2222',
53     'ivan.petrov@example.com',
54     '2019-06-15'
55 ),
56 (
57     'Olga',
58     'Sidorova',

```

```

59         1,
60         '+7-900-333-4444',
61         'olga.sidorova@example.com',
62         '2020-03-01'
63     ),
64     (
65         'Mikhail',
66         'Ivanov',
67         2,
68         '+7-900-555-6666',
69         'm.ivanov@example.com',
70         '2018-01-10'
71     ),
72     (
73         'Filipp',
74         'Kovalev',
75         3,
76         '+7-900-777-8888',
77         'f.kovalev@example.com',
78         '2021-05-20'
79     ) ON CONFLICT DO NOTHING;
80
81 -- =====
82 -- Клиенты (физические и корпоративные)
83 -- =====
84 INSERT INTO Client (
85     client_type_id,
86     first_name,
87     last_name,
88     company_name,
89     phone,
90     email,
91     address
92 )
93 VALUES (
94     1,
95     'Petr',
96     'Smirnov',
97     NULL,
98     '+7-912-000-1111',

```

```

99         'p.smirnov@example.com',
100         'Moscow, Tverskaya 1'
101     ),
102     (
103         2,
104         NULL,
105         NULL,
106         'TechSolutions LLC',
107         '+7-912-222-3333',
108         'sales@techsolutions.ru',
109         'Saint Petersburg, Nevsky 5'
110     ),
111     (
112         1,
113         'Anna',
114         'Kuznetsova',
115         NULL,
116         '+7-912-444-5555',
117         'anna.k@example.com',
118         'Kazan, Baumana 10'
119     ),
120     (
121         1,
122         'Dmitriy',
123         'Orlov',
124         NULL,
125         '+7-912-666-7777',
126         'd.orlov@example.com',
127         'Novosibirsk, Lenina 20'
128     ) ON CONFLICT DO NOTHING;
129
130 -- =====
131 -- Модели
132 -- =====
133 INSERT INTO Model (
134     brand,
135     model_name,
136     body_type_id,
137     engine_type_id,
138     transmission_id

```



```

139     )
140 VALUES ('Toyota', 'Camry', 1, 1, 2),
141         ('Porsche', 'Cayenne', 3, 2, 2),
142         ('Porsche', '911', 5, 1, 1),
143         ('BMW', 'M4', 1, 1, 2),
144         ('Tesla', 'Model 3', 4, 4, 2),
145         ('Audi', 'RS6', 6, 1, 2),
146         ('Skoda', 'Rapid', 1, 1, 1),
147         ('Volkswagen', 'Golf', 2, 1, 2),
148         ('Volkswagen', 'Touareg', 3, 1, 2) ON CONFLICT DO NOTHING;
149
150 -- =====
151 -- Автомобили
152 -- =====
153 INSERT INTO Car (
154     model_id,
155     vin,
156     year,
157     color,
158     price,
159     status_id
160 )
161 VALUES (
162     (
163         SELECT model_id
164         FROM model
165         WHERE brand = 'Toyota'
166             AND model_name = 'Camry'
167         LIMIT 1
168     ), 'JTNBB46KX20000001', 2021, 'Black', 3000000, 1
169 ), (
170     (
171         SELECT model_id
172         FROM model
173         WHERE brand = 'Porsche'
174             AND model_name = '911'
175         LIMIT 1
176     ), '2T3RFREV9FW000002', 2024, 'White', 28000000, 1
177 ), (
178     (

```

```

179         SELECT model_id
180         FROM model
181         WHERE brand = 'Porsche'
182               AND model_name = '911'
183         LIMIT 1
184     ), 'HP3RFRGEVOW004022', 2022, 'Silver', 25900000, 1
185 ), (
186     (
187         SELECT model_id
188         FROM model
189         WHERE brand = 'BMW'
190               AND model_name = 'M4'
191         LIMIT 1
192     ), 'WBA8E1C50GK000003', 2021, 'Blue', 9000000, 1
193 ), (
194     (
195         SELECT model_id
196         FROM model
197         WHERE brand = 'Audi'
198               AND model_name = 'RS6'
199         LIMIT 1
200     ), '1HGBH41JXMN109186', 2023, 'Gray', 12000000, 2
201 ), (
202     (
203         SELECT model_id
204         FROM model
205         WHERE brand = 'Skoda'
206               AND model_name = 'Rapid'
207         LIMIT 1
208     ), 'TRGLH34JZMU164106', 2017, 'White', 1050000, 4
209 ), (
210     (
211         SELECT model_id
212         FROM model
213         WHERE brand = 'Tesla'
214               AND model_name = 'Model 3'
215         LIMIT 1
216     ), 'TURBH31DSGN103158', 2023, 'White', 6000000, 1
217 ), (
218     (

```

```

219         SELECT model_id
220         FROM model
221         WHERE brand = 'Volkswagen'
222             AND model_name = 'Golf'
223         LIMIT 1
224     ), '3YGMH65JZOPU64146', 2018, 'Gray', 1800000, 1
225 ), (
226     (
227         SELECT model_id
228         FROM model
229         WHERE brand = 'Volkswagen'
230             AND model_name = 'Touareg'
231         LIMIT 1
232     ), 'RUDGL55JF00H54421', 2019, 'Black', 3950000, 1
233 ), (
234     (
235         SELECT model_id
236         FROM model
237         WHERE brand = 'Volkswagen'
238             AND model_name = 'Touareg'
239         LIMIT 1
240     ), 'UEGGH45JD0FU39046', 2020, 'Gray', 4900000, 1
241 ) ON CONFLICT DO NOTHING;
242
243 -- =====
244 -- Состояние автомобиля
245 -- =====
246 INSERT INTO CarCondition (
247     car_id,
248     mileage,
249     mechanical_condition,
250     body_condition,
251     interior_condition,
252     inspection_date,
253     notes
254 )
255 VALUES (
256     (
257         SELECT car_id
258         FROM car

```

```

259         WHERE vin = 'JTNBB46KX20000001'
260     ),
261     50000,
262     'Ok',
263     'Replaced bumper',
264     'Good',
265     '2025-11-01',
266     'Service done at 40000 km'
267 ),
268 (
269     (
270         SELECT car_id
271         FROM car
272         WHERE vin = '2T3RFREV9FW000002'
273     ),
274     112,
275     'Perfect',
276     'Perfect',
277     'Perfect',
278     '2025-10-15',
279     NULL
280 ),
281 (
282     (
283         SELECT car_id
284         FROM car
285         WHERE vin = 'HP3RFRGEVOW004022'
286     ),
287     9900,
288     'Perfect',
289     'Perfect',
290     'Perfect',
291     '2025-08-22',
292     NULL
293 ),
294 (
295     (
296         SELECT car_id
297         FROM car
298         WHERE vin = 'WBA8E1C50GK000003'

```

```

299         ),
300         32000,
301         'Engine goes tuk-tuk',
302         'Good',
303         'Good',
304         '2025-09-20',
305         'Engine needs check'
306     ),
307     (
308         (
309             SELECT car_id
310             FROM car
311             WHERE vin = 'TRGLH34JZMU164106'
312         ),
313         119000,
314         'Need technical service',
315         'Good',
316         'Need detailing',
317         '2025-10-23',
318         NULL
319     ),
320     (
321         (
322             SELECT car_id
323             FROM car
324             WHERE vin = 'TURBH31DSGN103158'
325         ),
326         27000,
327         'Strange sounds',
328         'Replaced bumper',
329         'Good',
330         '2025-05-25',
331         NULL
332     ),
333     (
334         (
335             SELECT car_id
336             FROM car
337             WHERE vin = '3YGMH65JZOPU64146'
338         ),

```

```

339         85000,
340         'Normal',
341         'Good',
342         'Need detailing',
343         '2025-09-02',
344         NULL
345     ),
346     (
347         (
348             SELECT car_id
349             FROM car
350             WHERE vin = 'RUDGL55JF00H54421'
351         ),
352         198000,
353         'Good',
354         'Good',
355         'Good',
356         '2025-08-29',
357         NULL
358     ),
359     (
360         (
361             SELECT car_id
362             FROM car
363             WHERE vin = 'UEGGH45JD0FU39046'
364         ),
365         84000,
366         'Good',
367         'Good',
368         'Not ideal',
369         '2025-07-12',
370         NULL
371     ) ON CONFLICT (car_id) DO
372 UPDATE
373 SET mileage = EXCLUDED.mileage,
374     mechanical_condition = EXCLUDED.mechanical_condition,
375     body_condition = EXCLUDED.body_condition,
376     interior_condition = EXCLUDED.interior_condition,
377     inspection_date = EXCLUDED.inspection_date,
378     notes = EXCLUDED.notes;

```

```

379
380 -- =====
381 -- Продажи
382 -- =====
383 INSERT INTO Sale (
384     sale_date,
385     car_id,
386     client_id,
387     employee_id,
388     sale_price
389 )
390 VALUES (
391     '2025-11-02',
392     (
393         SELECT car_id
394         FROM Car
395         WHERE vin = 'JTNBB46KX20000001'
396     ),
397     (
398         SELECT client_id
399         FROM Client
400         WHERE last_name = 'Smirnov'
401         LIMIT 1
402     ), (
403         SELECT employee_id
404         FROM Employee
405         WHERE last_name = 'Petrov'
406         LIMIT 1
407     ), 2930000
408 ), (
409     '2025-10-10', (
410         SELECT car_id
411         FROM Car
412         WHERE vin = 'WBA8E1C50GK000003'
413     ),
414     (
415         SELECT client_id
416         FROM Client
417         WHERE last_name = 'Kuznetsova'
418         LIMIT 1

```

```

419         ),(
420             SELECT employee_id
421             FROM Employee
422             WHERE last_name = 'Sidorova'
423             LIMIT 1
424         ), 9005000
425     ), (
426         '2025-10-25', (
427             SELECT car_id
428             FROM Car
429             WHERE vin = 'RUDGL55JF00H54421'
430         ),
431         (
432             SELECT client_id
433             FROM Client
434             WHERE company_name = 'TechSolutions LLC'
435             LIMIT 1
436         ), (
437             SELECT employee_id
438             FROM Employee
439             WHERE last_name = 'Petrov'
440             LIMIT 1
441         ), 3900000
442     ), (
443         '2025-11-01', (
444             SELECT car_id
445             FROM Car
446             WHERE vin = 'TRGLH34JZMU164106'
447         ),
448         (
449             SELECT client_id
450             FROM Client
451             WHERE company_name = 'TechSolutions LLC'
452             LIMIT 1
453         ), (
454             SELECT employee_id
455             FROM Employee
456             WHERE last_name = 'Ivanov'
457             LIMIT 1
458         ), 950000

```



```
) ON CONFLICT DO NOTHING;
```

Листинг 7: Заполнение тестовыми данными

Удаление всех созданных объектов представлено на листинге 8.

```

1  -- =====
2  -- Удалить процедуры
3  -- =====
4  DROP PROCEDURE IF EXISTS reserve_car(INT, INT, INT);
5
6  DROP PROCEDURE IF EXISTS refund_sale(INT, OUT INT, OUT INT, OUT INT);
7
8  -- =====
9  -- Удалить триггеры
10 -- =====
11 DROP TRIGGER IF EXISTS trg_update_stat_after_sale ON Sale;
12
13 DROP TRIGGER IF EXISTS trg_set_car_sold_after_sale ON Sale;
14
15 DROP TRIGGER IF EXISTS trg_maintain_car_condition ON Car;
16
17 DROP TRIGGER IF EXISTS trg_prevent_sale_of_sold_car ON Sale;
18
19 -- =====
20 -- Функции удаления
21 -- =====
22 DROP FUNCTION IF EXISTS update_employee_stat(INT);
23
24 DROP FUNCTION IF EXISTS get_inventory_json();
25
26 DROP FUNCTION IF EXISTS set_car_sold();
27
28 DROP FUNCTION IF EXISTS maintain_car_condition();
29
30 DROP FUNCTION IF EXISTS prevent_sale_of_sold_car();
31
32 -- =====
33 -- Удалить представления
34 -- =====
35 DROP VIEW IF EXISTS v_available_cars;
36
```

```

37 DROP VIEW IF EXISTS v_sales_report;
38
39 DROP VIEW IF EXISTS v_employee_performance;
40
41 -- =====
42 -- Удалить таблицы
43 -- =====
44 DROP TABLE IF EXISTS EmployeeStatistic CASCADE;
45
46 DROP TABLE IF EXISTS Sale CASCADE;
47
48 DROP TABLE IF EXISTS CarCondition CASCADE;
49
50 DROP TABLE IF EXISTS Car CASCADE;
51
52 DROP TABLE IF EXISTS Model CASCADE;
53
54 DROP TABLE IF EXISTS Client CASCADE;
55
56 DROP TABLE IF EXISTS Employee CASCADE;
57
58 DROP TABLE IF EXISTS StatusType CASCADE;
59
60 DROP TABLE IF EXISTS Position CASCADE;
61
62 DROP TABLE IF EXISTS TransmissionType CASCADE;
63
64 DROP TABLE IF EXISTS EngineType CASCADE;
65
66 DROP TABLE IF EXISTS BodyType CASCADE;
67
68 DROP TABLE IF EXISTS ClientType CASCADE;

```

Листинг 8: Удаление созданных объектов