

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

# Базы данных

Отчет по лабораторной работе №4  
Язык SQL-DML

**Работу**  
**выполнил:**  
Графов Д.И.  
Группа: 33531/2  
**Преподаватель:**  
Мяснов А.В.

Санкт-Петербург  
2019

# Содержание

<b>1. Цель работы</b>	<b>3</b>
<b>2. Программа работы</b>	<b>3</b>
<b>3. Теоретическая информация</b>	<b>3</b>
<b>4. Ход работы</b>	<b>3</b>
4.1. Выполнение запросов из списка стандартных запросов . . . . .	4
4.2. Выполнение индивидуального задания . . . . .	7
4.2.1. Индивидуальное задание №1 . . . . .	7
4.2.2. Индивидуальное задание №2 . . . . .	10
4.2.3. Индивидуальное задание №3 . . . . .	11
<b>5. Выводы</b>	<b>13</b>

## 1. Цель работы

Познакомить студентов с языком создания запросов управления данными SQL-DML.

## 2. Программа работы

- Изучение SQL-DML.
- Выполнение всех запросов из списка стандартных запросов. Демонстрация результатов преподавателю.
- Получение у преподавателя и реализация SQL-запросов в соответствии с индивидуальным заданием. Демонстрация результатов преподавателю.
- Сохранение в БД выполненных запросов SELECT в виде представлений, запросов INSERT, UPDATE или DELETE – в виде ХП. Выкладывание скрипта в GitLab.

## 3. Теоретическая информация

EXPLAIN Структура плана запроса представляет собой дерево узлов плана. Узлы на нижнем уровне дерева — это узлы сканирования, которые возвращают необработанные данные таблицы. Разным типам доступа к таблице соответствуют разные узлы: последовательное сканирование, сканирование индекса и сканирование битовой карты. Источниками строк могут быть не только таблицы, но и например, предложения VALUES и функции, возвращающие множества во FROM, и они представляются отдельными типами узлов сканирования. Если запрос требует объединения, агрегатных вычислений, сортировки или других операций с исходными строками, над узлами сканирования появляются узлы, обозначающие эти операции. И так как обычно операции могут выполняться разными способами, на этом уровне тоже могут быть узлы разных типов. В выводе команды EXPLAIN для каждого узла в дереве плана отводится одна строка, где показывается базовый тип узла плюс оценка стоимости выполнения данного узла, которую сделал для него планировщик. Если для узла выводятся дополнительные свойства, в вывод могут добавляться дополнительные строки, с отступом от основной информации узла. В самой первой строке (основной строке самого верхнего узла) выводится общая стоимость выполнения для всего плана; именно это значение планировщик старается минимизировать.

## 4. Ход работы

Каждая операция будет применена к одной таблице, так как для других выполнение будет аналогично.

## 4.1. Выполнение запросов из списка стандартных запросов

Листинг 1: common.sql

```
1  —выборка всех данных из таблицы
2  select *
3  from components;
4
5  —выборка данных из одной таблицы при нескольких условиях, с использованием логических операций
6  —LIKE, BETWEEN, IN
7  select *
8  from components
9  where title like 'водка';
10 select *
11 from drinks
12 where alcohol between 30 and 100;
13 select *
14 from food
15 where title in ('начос', 'стрипсы');
16
17 —вычисляемое поле в запросе
18 select count(drink_id)
19 from drinks;
20
21 —выборка всех данных с сортировкой по нескольким полям
22 select *
23 from places
24 order by address desc, place_id;
25
26 —запрос, вычисляющий несколько совокупных характеристик таблицы
27 select max(drink_id), min(place_id)
28 from drinks,
29      places;
30
31 —выборка данных из связанных таблиц
32 select d.title, p.title, p.address
33 from drinks d
34      join places_drinks pd on d.drink_id = pd.drink_id
35      join places p on p.place_id = pd.place_id
36 where d.title = 'боярский'
37 order by d.title desc;
38
39 —пример использования вложенного запроса
40 select dtitle
41 from (select drinks.title dtitle, c.title ctitle
42       from drinks
43       left outer join components_drinks cd on drinks.drink_id = cd.
44         ↪ drink_id
45       left outer join components c on cd.component_id = c.component_id)
46     ↪ as d
47 where d.ctitle = 'водка';
48
49 insert into components(title, alcohol)
50 values ('апельсиновый_сок', 0);
51
52 update components
53 set title = 'jagermeister',
54     alcohol = 35
55 where component_id = 5;
56
57 delete
```

```
56 from discounts
57 where discount_id = (select max(discount_id) from discounts);
58
59 delete
60 from places
61 where place_id not in (select place_id from supplies_drinks)
62 and place_id not in (select place_id from supplies_food);
```

Приведём некоторые примеры работы данного скрипта. Со всеми примерами можно ознакомиться, запустив скрипт, расположенный в репозитории.  
[http://gitlab.icc.spbstu.ru/grafa/wine\\_card](http://gitlab.icc.spbstu.ru/grafa/wine_card)

## Результаты

	component_id	title	alcohol
1	1	водка	40
2	2	гренадин	0
3	3	соус табаско	0
4	4	томатный сок	0
5	5	Hnm0ZwGdHlxrpFoQG8DxW9UwYgLJD	3
6	6	QiuS3N3KemVpgrUWiQkHw8LrnWYnwbF	44
7	7	aSNy5BHN0nTYnZoBJG4eA	12
8	8	aaFJNcdk82IhGPyJSid0rI1MRAqtND3F	6
9	9	jlEvt9kZpu	59
10	10	WwXA	19
11	11	RfUmCVLEqec2Qf7IzqlERbH0	35
12	12	BKKB2rIDt7	100
13	13	FeUsTRxqPkYujdWfKMixX80IXzM3f79phlNvbxod	93
14	14	R5ftxE3AXL8GQ72GfX0XTIX10JH01	53
15	15	7584nzPYUvhZZnPDIIy0rHxKyF09qybWkjzoJFD1vIw6hr5	68
16	16	4NzAfxQ	54
17	17	NaZfl4birZRMjS7DzC9MQNJFo	20
18	18	A7zxjl2D3vr6	81

Рисунок 4.1. Выборка всех данных из таблицы

	component_id	title	alcohol
1	1	водка	40

Рисунок 4.2. Оператор LIKE

	drink_id	title	rating	volume	alcohol	a
1	3	On6T4tqKT9UeZ0yVEjajJYH6Au0MPjfucCKJijl	3.2454765658217597	313	51	
2	5	j1KUxMg6BCVWH07ZKJsyphkLk9A2goQgeXJI0cH0SkLeNc	9.18238753785378	140	86	
3	6	I	2.1080099193866895	779	79	
4	8	7NkKE3LrtnhHsEgjCVDQClJYrfi	1.2459038924453414	874	53	
5	9	hent	1.9464013709911676	447	43	
6	10	Rfvz8KiBTqxf9aT4mubVx2QZ0NDCf0IVrfC	9.328979432074826	106	66	
7	11	l6	1.8325152005514858	432	35	
8	14	lPaFNW7A3Eh6opmQH8bZ8H	7.593012264319161	131	55	
9	15	NJi8r8GBnk5tcTjeh8uI4	8.221540510344944	623	83	
10	16	hi7zGA2ZJ8CV1y3hMDx5z8KmyQ5tAxmXW2xVrqKYPTT7FrFz	5.391543099045303	484	47	
11	18	qbxE3iIQZpezy3aak34p1JNIFENF8Tf5ipUfEHhAta0C	3.01594408552529356	751	95	
12	19	szHxmJk3vIOz	8.25180908496088	779	58	

Рисунок 4.3. Оператор BETWEEN

	food_id	title	rating	volume	average_price
1	1	стрипсы	9.5	<null>	<null>
2	2	начос	8	<null>	<null>

Рисунок 4.4. Оператор IN

## 4.2. Выполнение индивидуального задания

Мной было получено индивидуальное задание, состоящее из 3 частей:

1. Для каждого бара вывести топ-3 самых часто поставляемых за последний месяц единиц товара среди еды и напитков, рейтинг которых выше среднего по всем барам.
2. Вывести перечень напитков, у которых количество ингредиентов для приготовления больше среднего. Отсортировать по уменьшению градуса.
3. Для каждой акции вывести топ-3 напитков: самый крепкий алкогольный напиток, самый дешевый и самый большой по объему. Если для одной акции какие-то напитки в топ-3 совпадают, то выводить лишь один раз, без дублирования.

Для всех запросов необходимо привести план запроса и пояснения.

### 4.2.1. Индивидуальное задание №1

Структурно запрос можно разделить на 2 части: первая вычисляет требуемую выборку для напитков, вторая – для еды. Рассмотрим работу одного из подзапросов:

С помощью оператора `join` мы формируем выборку из MANY-MANY таблицы `supplies_drinks` и связанных с ней `drinks` и `places`. Затем с помощью оператора `where ... and ...` проводим фильтрацию по дате поставки (за последний месяц) и по рейтингу.

Затем вычисляем оконную функцию

*`row_number() over (partition by p.place_id order by d.drink_id) top`*

для подсчёта количества напитков в каждом заведении.

Далее из получившегося запроса делаем подзапрос и с помощью условия *`where top <= 3`* получаем топ-3 напитка для каждого бара с заданными условиями.

## Листинг 2: individual\_1.sql

```

1  —ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
2  —1
3  (select *
4  from (select distinct p.place_id ,
5           p.title
6           ↪      place_title ,
7           p.address ,
8           p.rating
9           ↪      place_rating ,
10          d.drink_id
11          ↪      item_id ,
12          d.title
13          ↪      item_title ,
14          d.rating
15          ↪      item_rating ,
16          sd.date ,
17          row_number() over (partition by p.place_id order by d.
18          ↪ drink_id) top
19      from supplies_drinks sd
20      join drinks d on sd.drink_id = d.drink_id
21      join places p on sd.place_id = p.place_id
22      where date > (select (now() - interval '1_month'))
23      and d.rating > (select avg(rating) from drinks)) as subquery
24 where top <= 3)
25 union
26 (select *
27 from (select distinct p.place_id ,
28          p.title
29          ↪      place_title ,
30          p.address ,
31          p.rating
32          ↪      place_rating ,
33          f.food_id
34          ↪      item_id ,
35          f.title
36          ↪      item_title ,
37          f.rating
38          ↪      item_rating ,
39          sd.date ,
40          row_number() over (partition by p.place_id order by f.
41          ↪ food_id) top
42      from supplies_food sd
43      join food f on sd.food_id = f.food_id
44      join places p on sd.place_id = p.place_id
45      where date > (select (now() - interval '1_month'))
46      and f.rating > (select avg(rating) from drinks)) as subquery
47 where top <= 3)
48 order by place_id , item_id;
```

Ниже приведён план первого подзапроса (до union), вычисленный с помощью оператора EXPLAIN ANALYSE. План запроса для второй части запроса аналогичен.

## Листинг 3: План запроса

```

1 Subquery Scan on subquery  (cost=3338.03..3588.05 rows=3704 width=124) (actual
2   ↪ time=32.642..32.830 rows=202 loops=1)
3 Filter: (subquery.top <= 3)
4 Rows Removed by Filter: 32
```



```

4 -> HashAggregate (cost=3338.03..3449.15 rows=11112 width=124) (actual time
    ↳ =32.640..32.800 rows=234 loops=1)
5 Group Key: p.place_id, p.title, p.address, p.rating, d.drink_id, d.title, d.
    ↳ rating, sd.date, row_number() OVER (?)
6 InitPlan 1 (returns $0)
7 -> Result (cost=0.00..0.01 rows=1 width=8) (actual time=0.004..0.004 rows=1
    ↳ loops=1)
8 InitPlan 2 (returns $1)
9 -> Aggregate (cost=3.28..3.29 rows=1 width=8) (actual time=0.045..0.045 rows=1
    ↳ loops=1)
10 -> Seq Scan on drinks (cost=0.00..3.02 rows=102 width=8) (actual time
    ↳ =0.005..0.018 rows=102 loops=1)
11 -> WindowAgg (cost=2862.47..3084.71 rows=11112 width=124) (actual time
    ↳ =32.305..32.448 rows=234 loops=1)
12 -> Sort (cost=2862.47..2890.25 rows=11112 width=116) (actual time
    ↳ =32.297..32.313 rows=234 loops=1)
13 Sort Key: p.place_id, d.drink_id
14 Sort Method: quicksort Memory: 73kB
15 -> Hash Join (cost=8.20..2115.75 rows=11112 width=116) (actual time
    ↳ =0.382..32.075 rows=234 loops=1)
16 Hash Cond: (sd.place_id = p.place_id)
17 -> Hash Join (cost=3.70..2080.97 rows=11112 width=53) (actual time
    ↳ =0.284..31.801 rows=234 loops=1)
18 Hash Cond: (sd.drink_id = d.drink_id)
19 -> Seq Scan on supplies_drinks sd (cost=0.00..1986.12 rows=33337 width=16) (
    ↳ actual time=0.047..31.304 rows=426 loops=1)
20 Filter: (date > $0)
21 Rows Removed by Filter: 99584
22 -> Hash (cost=3.28..3.28 rows=34 width=41) (actual time=0.229..0.229 rows=54
    ↳ loops=1)
23 Buckets: 1024 Batches: 1 Memory Usage: 13kB
24 -> Seq Scan on drinks d (cost=0.00..3.28 rows=34 width=41) (actual time
    ↳ =0.056..0.081 rows=54 loops=1)
25 Filter: (rating > $1)
26 Rows Removed by Filter: 48
27 -> Hash (cost=3.11..3.11 rows=111 width=67) (actual time=0.090..0.090 rows=110
    ↳ loops=1)
28 Buckets: 1024 Batches: 1 Memory Usage: 18kB
29 -> Seq Scan on places p (cost=0.00..3.11 rows=111 width=67) (actual time
    ↳ =0.014..0.046 rows=110 loops=1)
30 Planning time: 0.596 ms
31 Execution time: 33.005 ms

```

place_id	place_title	address	place_rating	item_id	item_title	item_rating	date	top
1	12	road11e0rAgTdrIE2uIn31jnkSpYPeT0STnz	Avs32MA5N	19	szhmjJk3v10z	8.25189989496088	2019-05-03 09:49:16.607492	1
2	12	road11e0rAgTdrIE2uIn31jnkSpYPeT0STnz	Avs32MA5N	20	4bor9PMRrZQW15qE3ovosv2R8H9QmB24q5Ph	9.2709980348339696	2019-05-03 20:28:13.319667	2
3	13	F18APXkuhFzpusVmwovvB8NSJwT18AakfgerHLLI	5cxhbg3bvxxeXI1ApJ70CEdbIqPLJNDY0B15FU	48	akR9Wlj24xFp7dujyhm0	7.9377052489492	2019-05-30 02:29:14.138629	1
4	13	F18APXkuhFzpusVmwovvB8NSJwT18AakfgerHLLI	5cxhbg3bvxxeXI1ApJ70CEdbIqPLJNDY0B15FU	90	AZL4yFSYE1DeGtWTUdCp	8.541402616877896	2019-05-06 03:53:23.186458	2
5	13	F18APXkuhFzpusVmwovvB8NSJwT18AakfgerHLLI	5cxhbg3bvxxeXI1ApJ70CEdbIqPLJNDY0B15FU	108	5w0bHL	9.23673344952968	2019-06-01 16:53:29.232918	1
6	14	H06C0Gpw5t13z1f67	gwCX	7	e3jnc1bdi29gb5xPAITzg9	9.40390231532688	2019-05-28 23:58:30.842160	1
7	14	H06C0Gpw5t13z1f67	gwCX	14	lPaFW7AJEH6opmQ8bZ8H	7.593012264319161	2019-05-14 05:46:29.720841	1
8	14	H06C0Gpw5t13z1f67	gwCX	53	q8Bew5Q0W0spno5dI	6.441918909488162	2019-05-03 19:23:09.549124	3
9	15	56NGoc9ASu0tRXSQ0MI	1bw4xt95sdP7UH	48	Z2wqlyt1Dgnocn9mGhnxpY180Ygoc7ki8D	9.547188639832985	2019-05-28 01:02:16.216047	1
10	15	56NGoc9ASu0tRXSQ0MI	1bw4xt95sdP7UH	79	oP7A2740J	7.178292282493561	2019-05-31 02:10:11.055087	1
11	16	TLPeas1fu0tPBg8quT24VrHn1KycTDLNupysUgR4pcyT	1vW7zm	17	roshS1lo0Isal3xd9gCQGLKjVTC8d1uLeRDWHR5Nw	9.38443519829346	2019-05-18 21:35:11.054260	1
12	16	TLPeas1fu0tPBg8quT24VrHn1KycTDLNupysUgR4pcyT	1vW7zm	70	rVY3RhrhaapRHZqFfTHkBgJMD0DF816D4G01yhsyr106I	6.2503873612993495	2019-05-15 11:46:24.112158	2
13	17	TuK8dCjMDV0Rog2ehontzTfYcyU0NRupgDAG5	UvTQv4kGBNTRK2QrsqfUyUgHw4NM6Asu0a1oUC7	57	zmK2HsNKLdtdapLy9H5Q2b6gOFFFA3F5e8H	8.98075957506955	2019-06-01 11:25:11.680765	1
14	17	TuK8dCjMDV0Rog2ehontzTfYcyU0NRupgDAG5	UvTQv4kGBNTRK2QrsqfUyUgHw4NM6Asu0a1oUC7	59	WAEf6Ew0dcxvEChjG1K6G05hK2rE	7.866210478456251	2019-05-15 02:36:42.078185	2
15	17	TuK8dCjMDV0Rog2ehontzTfYcyU0NRupgDAG5	UvTQv4kGBNTRK2QrsqfUyUgHw4NM6Asu0a1oUC7	91	4FH9hqs1p4yVYhbmVcFa8LZY8Mx7ybjsvvpPB0A8p	9.70942153078607	2019-05-08 17:19:21.115841	3
16	18	AKA52u8e3RgAAb1VLVe0n5d9k34kP20lg	Vsc3q2bYf0TcfsTj1b000PrPZKsDA	43	3APJ3JkD6gUgXVub5eH7	7.8551042691157935	2019-05-12 16:27:26.036176	1
17	18	AKA52u8e3RgAAb1VLVe0n5d9k34kP20lg	Vsc3q2bYf0TcfsTj1b000PrPZKsDA	51	p5vq5n1d4g5K1PLFgtTNA05DFrK7N28awvVf	8.403238923519082	2019-05-18 23:13:17.731287	2
18	18	AKA52u8e3RgAAb1VLVe0n5d9k34kP20lg	Vsc3q2bYf0TcfsTj1b000PrPZKsDA	57	zmK2HsNKLdtdapLy9H5Q2b6gOFFFA3F5e8H	8.98075957506955	2019-05-18 09:27:52.016515	1
19	21	1qyo1g829WfWZe1dH8sKd9d736HrG	ajwayT	33	eiqU229Eg7Rhx3Y81TP0av1j0KkgafKdmuqUkP09VrQf8	7.83268735743788	2019-05-19 22:10:39.422579	1
20	21	1qyo1g829WfWZe1dH8sKd9d736HrG	ajwayT	42	2LLD0v1Y5HmRpxPLWMBjBKr3TDXpku3Vn8t	7.444373525027923	2019-06-03 03:38:53.154078	2
21	21	1qyo1g829WfWZe1dH8sKd9d736HrG	ajwayT	68	3BVM16v	6.928426192911285	2019-05-21 10:30:38.893200	2
22	23	5k1ND6k3KdEF0wELFCPwJx20tEmK1yPp4j2337VIE	skjcs081s50UQMHQPNV2JgFm9H8dKID9y2NBWwA1YY	51	p5vq5n1d4g5K1PLFgtTNA05DFrK7N28awvVf	8.403238923519082	2019-06-02 23:27:30.535862	1
23	23	5k1ND6k3KdEF0wELFCPwJx20tEmK1yPp4j2337VIE	skjcs081s50UQMHQPNV2JgFm9H8dKID9y2NBWwA1YY	73	Bm186E8JYR8BoTfpvR8xFe6GLV0uLe0	7.42373197250359	2019-05-28 21:35:22.679137	2
24	23	5k1ND6k3KdEF0wELFCPwJx20tEmK1yPp4j2337VIE	skjcs081s50UQMHQPNV2JgFm9H8dKID9y2NBWwA1YY	91	4FH9hqs1p4yVYhbmVcFa8LZY8Mx7ybjsvvpPB0A8p	9.70942153078607	2019-05-08 17:19:21.115841	3
25	24	VLfCv2Rw0mhtg2ghV0MgTAR05nfxLLIdfgwa90H1K377ss9	Q6wsQ7kebeY3DfH5PH1q5Lg181X4yRwZTLahMk14vzdc9	42	2LLD0v1Y5HmRpxPLWMBjBKr3TDXpku3Vn8t	7.444373525027923	2019-05-18 03:31:59.734340	1
26	24	VLfCv2Rw0mhtg2ghV0MgTAR05nfxLLIdfgwa90H1K377ss9	Q6wsQ7kebeY3DfH5PH1q5Lg181X4yRwZTLahMk14vzdc9	48	Z2wqlyt1Dgnocn9mGhnxpY180Ygoc7ki8D	9.547188639832985	2019-05-28 01:02:16.216047	1
27	24	VLfCv2Rw0mhtg2ghV0MgTAR05nfxLLIdfgwa90H1K377ss9	Q6wsQ7kebeY3DfH5PH1q5Lg181X4yRwZTLahMk14vzdc9	76	51GmZ8B8Fh1C8K7p8e0LB88H	7.98802055853325	2019-05-14 06:04:47.688861	3
28	25	ahp1AJ11pnlk1f4y	tcH2MCmCwCjA183p19LFIj1x4u5A9t	46	akR9Wlj24xFp7dujyhm0	7.9377052489492	2019-05-04 07:57:03.379952	1

Рисунок 4.5. Результат работы

#### 4.2.2. Индивидуальное задание №2

Сначала формируется выборка из пар значений типа напитков – количество компонентов.

```
(select d.drink_id, count(d.drink_id) components from drinks d join components_drinks
cd on d.drink_id = cd.drink_id group by d.drink_id) sub
```

Затем получившаяся выбока присоединяется к таблице напитков по равенству drink\_id. Далее вычисляем среднее количество компонентов среди всех напитков и ограничиваем выборку с помощью оператора where. Наконец, группируем по drink\_id, components и сортируем по убыванию значения alcohol.

Листинг 4: individual\_2.sql

```
1 —ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
2 —2
3 select *
4 from (select d.drink_id, count(d.drink_id) components
5       from drinks d
6       join components_drinks cd on d.drink_id = cd.drink_id
7       group by d.drink_id) sub
8       join drinks d on d.drink_id = sub.drink_id
9 where components > (select avg(cnt)
10                    from (select count(d.drink_id) cnt
11                          from drinks d
12                          join components_drinks cd on d.drink_id = cd.
13                             ↪ drink_id
14                             group by d.drink_id) sub)
15 group by d.drink_id, sub.drink_id, sub.components
16 order by alcohol desc;
```

Листинг 5: План запроса

```
1 Sort (cost=51.88..52.13 rows=102 width=97) (actual time=1.113..1.116 rows=50
  ↪ loops=1)
2 Sort Key: d.alcohol DESC
3 Sort Method: quicksort Memory: 34kB
4 InitPlan 1 (returns $0)
5 -> Aggregate (cost=20.48..20.49 rows=1 width=32) (actual time=0.468..0.468
  ↪ rows=1 loops=1)
6 -> HashAggregate (cost=18.19..19.21 rows=102 width=12) (actual time
  ↪ =0.427..0.446 rows=102 loops=1)
7 Group Key: d_2.drink_id
8 -> Hash Join (cost=4.29..15.40 rows=558 width=4) (actual time=0.046..0.262
  ↪ rows=558 loops=1)
9 Hash Cond: (cd_1.drink_id = d_2.drink_id)
10 -> Seq Scan on components_drinks cd_1 (cost=0.00..9.58 rows=558 width=4) (
  ↪ actual time=0.006..0.066 rows=558 loops=1)
11 -> Hash (cost=3.02..3.02 rows=102 width=4) (actual time=0.035..0.035 rows=102
  ↪ loops=1)
12 Buckets: 1024 Batches: 1 Memory Usage: 12kB
13 -> Seq Scan on drinks d_2 (cost=0.00..3.02 rows=102 width=4) (actual time
  ↪ =0.004..0.018 rows=102 loops=1)
14 -> HashAggregate (cost=26.96..27.98 rows=102 width=97) (actual time
  ↪ =1.049..1.065 rows=50 loops=1)
15 Group Key: d.drink_id, d_1.drink_id, (count(d_1.drink_id))
```

```

16 -> Hash Join (cost=22.90..26.19 rows=102 width=81) (actual time=0.985..1.023
    ↳ rows=50 loops=1)
17 Hash Cond: (d.drink_id = d_1.drink_id)
18 -> Seq Scan on drinks d (cost=0.00..3.02 rows=102 width=69) (actual time
    ↳ =0.007..0.017 rows=102 loops=1)
19 -> Hash (cost=21.62..21.62 rows=102 width=12) (actual time=0.971..0.971 rows
    ↳ =50 loops=1)
20 Buckets: 1024 Batches: 1 Memory Usage: 11kB
21 -> HashAggregate (cost=19.58..20.60 rows=102 width=12) (actual time
    ↳ =0.920..0.960 rows=50 loops=1)
22 Group Key: d_1.drink_id
23 Filter: ((count(d_1.drink_id))::numeric > $0)
24 Rows Removed by Filter: 52
25 -> Hash Join (cost=4.29..15.40 rows=558 width=4) (actual time=0.057..0.267
    ↳ rows=558 loops=1)
26 Hash Cond: (cd.drink_id = d_1.drink_id)
27 -> Seq Scan on components_drinks cd (cost=0.00..9.58 rows=558 width=4) (actual
    ↳ time=0.015..0.075 rows=558 loops=1)
28 -> Hash (cost=3.02..3.02 rows=102 width=4) (actual time=0.038..0.038 rows=102
    ↳ loops=1)
29 Buckets: 1024 Batches: 1 Memory Usage: 12kB
30 -> Seq Scan on drinks d_1 (cost=0.00..3.02 rows=102 width=4) (actual time
    ↳ =0.004..0.019 rows=102 loops=1)
31 Planning time: 0.496 ms
32 Execution time: 1.225 ms

```

drink_id	components	drink_id	title	rating	volume	alcohol	average_price	drink_type
1	101	7	101 o995ayskTQwdVGEjonoUyHhKuG1AbiRskNc6	1.9113625924617534	237	100	705	шоты
2	49	8	49 3G084wvvySz7ZFelsfpgmgjvh	4.992381553203634	967	99	432	пиво и сидр
3	15	7	15 NJ18r8GBnk5tcTjeh8uI4	8.221540510344944	623	83	720	биттеры
4	65	7	65 yFBAjvk1BPezTt5JDkWKfJES5pdk	1.0786376037317726	979	83	436	пиво и сидр
5	27	6	27 tgD8r	1.5816964014740487	726	82	851	текила
6	99	7	99 8JRpkgACAlvxPjIE38dzfyATj3o5BUCjFDMGslz8Cm	3.8457526077164186	320	80	400	шоты
7	37	10	37 Ew82IstL4F0bzWRoagZA8ni6K4D1L0dHHEbi	6.05203362224787	895	80	777	ром
8	6	9	6 I	2.1988099193866895	779	79	916	ром
9	66	8	66 Kay0Fk2jhxqLH9zygkcPXgPYR3guj17n6fL	4.782681197396917	750	77	329	бренди и коньяк
10	26	10	26 XkqlyX367XXvWartE8U6BvTRYXMac8	8.243115815152578	132	76	843	биттеры
11	93	7	93 yu	4.957555294701924	146	75	683	бренди и коньяк
12	45	10	45 JZ0FlfoLroJyHpQ1KwE8UzH	4.423233602449926	688	65	209	текила
13	40	7	40 akR9Wlj24xFaf7dwjyhm0	7.9377052489492	745	64	311	коктейли
14	85	8	85 nKPZ0ECT	7.425067627972978	171	63	140	текила
15	58	7	58 dja2h3X3anFy3WqkfzEtCmy	6.382778480996955	791	63	853	водка и настойки
16	69	6	69 T1iLLqzjFRQ1XLlnKPD1R7UsI6xxhp4Uxt	7.823626203741262	184	63	967	коктейли
17	20	7	20 wbor9H9KkRrz0Wn15quEJoovsv2R8H9QmB24qSPH	9.270990348339696	284	63	432	безалкогольные
18	19	7	19 szHmJjK3vIOZ	8.251809084960888	779	58	405	пиво и сидр
19	31	8	31 dFNP0vm7Musfuka0trTNG7ybB0Hdm4DT6Tt2BNHzb	9.819890808892416	617	55	935	виски
20	8	7	8 7NKK3LrtnHsEgJCvdQC1JYrfi	1.2459038924453414	874	53	525	пиво и сидр
21	97	6	97 seG7yL403yzk7WkNyojI	1.965267472606928	402	52	828	ром
22	3	10	3 On6T4tqT9UeZ0yVEjajJYH6Au0MPj fucCKJijl	3.2454765656217597	313	51	991	пиво и сидр
23	98	7	98 7VkdEvJ6LqJqC646tyCvrgG1WkNjt5ZgJrgoAC7JNoMKMsI	1.8822093360745336	288	47	582	вино
24	78	10	78 CmwDKKcooIuLxxDU1Db8FRZu8mntTx2Vez74111ZADtq1B	6.542531756313932	947	46	202	биттеры
25	100	8	100 SmbBH1	9.2367333440952968	142	43	295	биттеры
26	9	10	9 hent	1.9464013709911676	447	43	155	бренди и коньяк
27	67	8	67 jwE8NjdPMSu7u5H	6.9840412399588	355	41	714	текила
28	54	8	54 Ww3IIS0XihCGwri0A2pwqmsKN3q5ejkpoyhyaQl6H4N7t6f	9.830799926371498	193	37	830	водка и настойки
29	11	8	11 l6	1.8325152005514858	432	35	578	бренди и коньяк

Рисунок 4.6. Результат работы

#### 4.2.3. Индивидуальное задание №3

С помощью оператора JOIN происходит объединение таблиц discounts и drinks по равенству поля drink\_type. Таким образом, получем выборку из всевозможных акций на напитки. Далее получившуюся выбоку ограничиваем с помощью оператора where ... or ... or .. и требуемых условий.

### Листинг 6: individual\_3.sql

```

1  —ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
2  —3
3  select *
4  from discounts
5      join drinks on drinks.drink_type = discounts.drink_type
6  where alcohol = (select max(alcohol) from drinks)
7      or average_price = (select min(average_price) from drinks)
8      or volume = (select max(volume) from drinks)
9  group by discount_id, drink_id
10 order by alcohol desc , average_price, volume desc;

```

### Листинг 7: План запроса

```

1  Sort (cost=69.02..69.94 rows=365 width=135) (actual time=1.689..1.726 rows=420
   ↳ loops=1)
2  Sort Key: drinks.alcohol DESC, drinks.average_price, drinks.volume DESC
3  Sort Method: quicksort Memory: 136kB
4  InitPlan 1 (returns $0)
5  -> Aggregate (cost=3.27..3.28 rows=1 width=8) (actual time=0.045..0.045 rows=1
   ↳ loops=1)
6  -> Seq Scan on drinks drinks_1 (cost=0.00..3.02 rows=102 width=8) (actual time
   ↳ =0.004..0.020 rows=102 loops=1)
7  InitPlan 2 (returns $1)
8  -> Aggregate (cost=3.27..3.28 rows=1 width=8) (actual time=0.041..0.041 rows=1
   ↳ loops=1)
9  -> Seq Scan on drinks drinks_2 (cost=0.00..3.02 rows=102 width=8) (actual time
   ↳ =0.004..0.018 rows=102 loops=1)
10 InitPlan 3 (returns $2)
11 -> Aggregate (cost=3.27..3.28 rows=1 width=8) (actual time=0.038..0.038 rows=1
   ↳ loops=1)
12 -> Seq Scan on drinks drinks_3 (cost=0.00..3.02 rows=102 width=8) (actual time
   ↳ =0.004..0.017 rows=102 loops=1)
13 -> HashAggregate (cost=39.99..43.64 rows=365 width=135) (actual time
   ↳ =1.016..1.239 rows=420 loops=1)
14 Group Key: discounts.discount_id, drinks.drink_id
15 -> Hash Join (cost=3.83..38.16 rows=365 width=135) (actual time=0.203..0.709
   ↳ rows=420 loops=1)
16 Hash Cond: (discounts.drink_type = drinks.drink_type)
17 -> Seq Scan on discounts (cost=0.00..23.10 rows=1010 width=66) (actual time
   ↳ =0.011..0.172 rows=1009 loops=1)
18 -> Hash (cost=3.79..3.79 rows=4 width=69) (actual time=0.164..0.164 rows=4
   ↳ loops=1)
19 Buckets: 1024 Batches: 1 Memory Usage: 9kB
20 -> Seq Scan on drinks (cost=0.00..3.79 rows=4 width=69) (actual time
   ↳ =0.138..0.161 rows=4 loops=1)
21 Filter: ((alcohol = $0) OR (average_price = $1) OR (volume = $2))
22 Rows Removed by Filter: 98
23 Planning time: 0.722 ms
24 Execution time: 1.853 ms

```

Table - d (TSV) View Query														
discount_id	place_id	drink_type	amount	description	weekday	time_start	time_end	drink_id	title	rating	volume	alcohol	average_price	drink_type
1	624	39 шоты	0.52541042622153	Asbncmp5D1i06GtUnL	<null>	17:17:11	07:40:40	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
2	76	79 шоты	0.11074455257338668	fF1a1nR5UFxNFOL0	<null>	18:05:17	20:41:23	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
3	393	75 шоты	0.10227035312994148	j74	<null>	21:14:06	03:45:53	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
4	113	53 шоты	0.2042641442429362	3nXPtY3u2kgJw02IL	<null>	21:44:00	15:39:18	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
5	562	42 шоты	0.1020231712048120	yWjWkV4ToB8au6S	<null>	02:53:37	10:10:09	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
6	804	98 шоты	0.5572954247086359	ShznqldpyIE7aPe1rnk	<null>	12:20:46	22:00:25	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
7	886	24 шоты	0.811107921026346	fEcsFuejBeeVYJvFWL	<null>	07:26:35	16:44:15	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
8	563	88 шоты	0.07702560804120462	EF1zgjKekKekKIPoTg	<null>	23:09:23	01:28:19	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
9	570	106 шоты	0.75239279738947	Dgd3i0ggyeKf1JmK07	<null>	03:30:50	05:21:46	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
10	77	89 шоты	0.7140478747375241	ddmKRCpPfeY0qfVLH	<null>	02:37:56	19:20:31	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
11	134	93 шоты	0.10849449970408154	LSyKc0JmZgdvrolm5K	<null>	05:05:27	13:55:12	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
12	797	39 шоты	0.20287994764640776	GJX53Pm	<null>	06:35:39	12:03:20	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
13	880	65 шоты	0.547645458254855	YjAmVneCedi05AKAjb	<null>	21:19:40	14:20:41	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
14	12	64 шоты	0.591633940134515	lARuqoig18Y00K	<null>	12:14:57	10:36:19	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
15	351	75 шоты	0.3642066775525209	hM0J0i1PhKvYzkeZ	<null>	01:15:42	15:36:12	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
16	496	49 шоты	0.33142285142203175	IJfIT1Y9	<null>	16:03:44	02:10:56	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
17	90	47 шоты	0.9778625500566958	FuECMcMD9mqb4P	<null>	06:39:05	08:15:46	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
18	11	56 шоты	0.086486943778748366	za9YumKfK7bZ6Z6	<null>	19:01:36	07:12:31	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
19	543	17 шоты	0.4015437120384085	kX0m8nSH2A	<null>	11:11:05	10:23:30	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
20	603	40 шоты	0.21418407826782804	z6gKhY5GmaLFAATckh	<null>	08:38:06	23:44:56	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
21	501	35 шоты	0.2677678255534154	h0BvU8B3pKXASykd1	<null>	10:42:39	05:34:55	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
22	40	59 шоты	0.0444749901795077	J51ppm0755	<null>	21:50:03	04:50:51	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
23	556	94 шоты	0.953209206635237	U23YvMdnQeGRCajZ	<null>	12:49:54	05:02:35	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
24	157	16 шоты	0.19653361450647933	2r1x9199K1BEcy52q	<null>	03:53:46	02:55:13	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
25	928	24 шоты	0.9148839282698943	M01f2F96Q0dTMcnovL	<null>	00:30:20	04:06:30	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
26	714	50 шоты	0.9190179504209279	oXN0K13AF2RFZ551p	<null>	02:29:23	22:03:54	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
27	162	69 шоты	0.5033579814067536	ZpUcXct0eN	<null>	06:23:00	20:50:32	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
28	98	94 шоты	0.8585202211397952	H7zvFRW6Lc06Z6b6bbL	<null>	00:43:39	03:32:32	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	
29	485	32 шоты	0.3603577575758333	tv8j072Z	<null>	11:05:36	19:07:23	101	0995aysKTQwdVGEjOnuohyKug1Ab1RsKnc6	1.9113625924617534	237	100	705 шоты	

Рисунок 4.7. Результат работы

## 5. Выводы

В ходе работы были изучен язык SQL-DML. Были созданы запросы к базе данных с использованием операторов выбора SELECT, INSERT, DELETE и UPDATE.

Для извлечения записей из таблиц в SQL определен оператор SELECT. Этот оператор возвращает ни одного, одно или множество строк, удовлетворяющих указанному условию и упорядоченных по заданному критерию.

Оператор SELECT позволяет возвращать не только множество значений полей, но и некоторые совокупные (агрегированные) характеристики, подсчитанные по всем или по указанным записям таблицы, например, SUM (<имя поля>) – сумма всех значений данного поля.

Также были рассмотрены и проанализированы планы запросов.