

Анализ данных о популярности ресторанов

Никишин Владимир Игоревич

Описание датасета

Dataset: <https://www.kaggle.com/datasets/new-york-city/nyc-inspections>

Датасет содержит данные о нарушениях и принятых по ним инспекциях в ресторанах Нью-Йорка.

Цель проекта:

Исследование факторов, влияющих на популярность ресторанов в Нью-Йорке

```
camis text,  
dba text,  
boro text,  
building text,  
street text,  
zipcode text,  
phone text,  
cuisine_description text,  
inspection_date date,  
action text,  
violation_code text,  
violation_description text,  
critical_flag text,  
score integer,  
grade text,  
grade_date date,  
record_date date,  
inspection_type text
```

Предобработка данных

select * from restaurants where extract(year from inspection_date) = 1900

restaurants 1 X

select * from restaurants where extract(year from inspectic Введите SQL выражение чтобы отфильтровать результаты

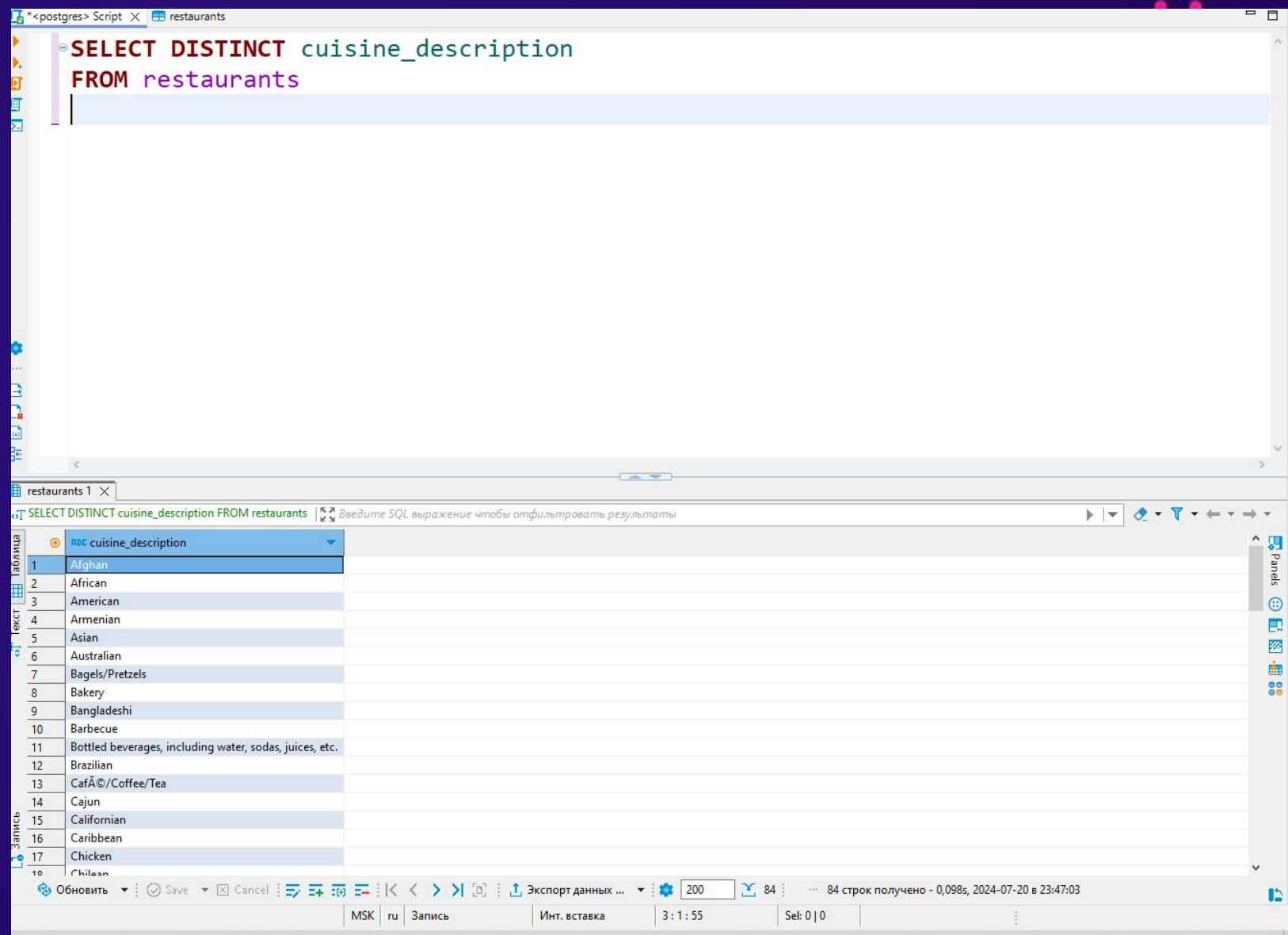
	camis	dba	boro	building	street	zipcode	phone	cuisine_description	inspection_date	action	violation_code
1	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
2	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
3	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
4	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
5	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
6	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
7	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
8	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
9	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
10	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
11	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
12	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
13	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
14	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
15	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
16	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
17	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
18	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
19	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
20	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
21	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
22	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		
23	50 067 644	NIEVES TIA MIMI ICEES	QUEENS	8914	ROOSEVELT AVE	11372	9 172 579 930	Other	1900-01-01		
24	50 067 671	GINA	MANHATTAN	2028	BROADWAY	10023	2 125 956 900	Other	1900-01-01		
25	50 066 251	KINI	QUEENS	4227	35TH AVE	11101	2 019 687 446	Other	1900-01-01		

Обновить Save Cancel Экспорт данных ... 200 120 120 строк получено - 0,004s, 2024-07-20 в 23:04:20

MSK ru Запись Инт. вставка 2 : 1 : 75 Sel: 0 | 0

SQL- запрос

- Запрос 1: Выведите список всех типов кухонь ресторанов в базе данных.



The screenshot displays a PostgreSQL client window with a script editor and a results pane. The script editor contains the following SQL query:

```
SELECT DISTINCT cuisine_description  
FROM restaurants
```

The results pane shows the output of the query, which is a list of unique cuisine descriptions. The results are displayed in a table with two columns: a row number and the cuisine description.

	cuisine_description
1	Afghan
2	African
3	American
4	Armenian
5	Asian
6	Australian
7	Bagels/Pretzels
8	Bakery
9	Bangladeshi
10	Barbecue
11	Bottled beverages, including water, sodas, juices, etc.
12	Brazilian
13	Café/Coffee/Tea
14	Cajun
15	Californian
16	Caribbean
17	Chicken
18	Chilean

The status bar at the bottom indicates that 84 rows were retrieved in 0.098 seconds on 2024-07-20 at 23:47:03.

SQL- запрос

- Запрос 2:
Выведите количество ресторанов, инспектированных в каждом году, отсортированное по убыванию.

The screenshot shows a PostgreSQL client interface with a script editor and a results pane. The script editor contains the following SQL query:

```
SELECT EXTRACT(YEAR FROM inspection_date) AS year, COUNT(*) AS inspection_count
FROM restaurants
GROUP BY year
ORDER BY inspection_count DESC;
```

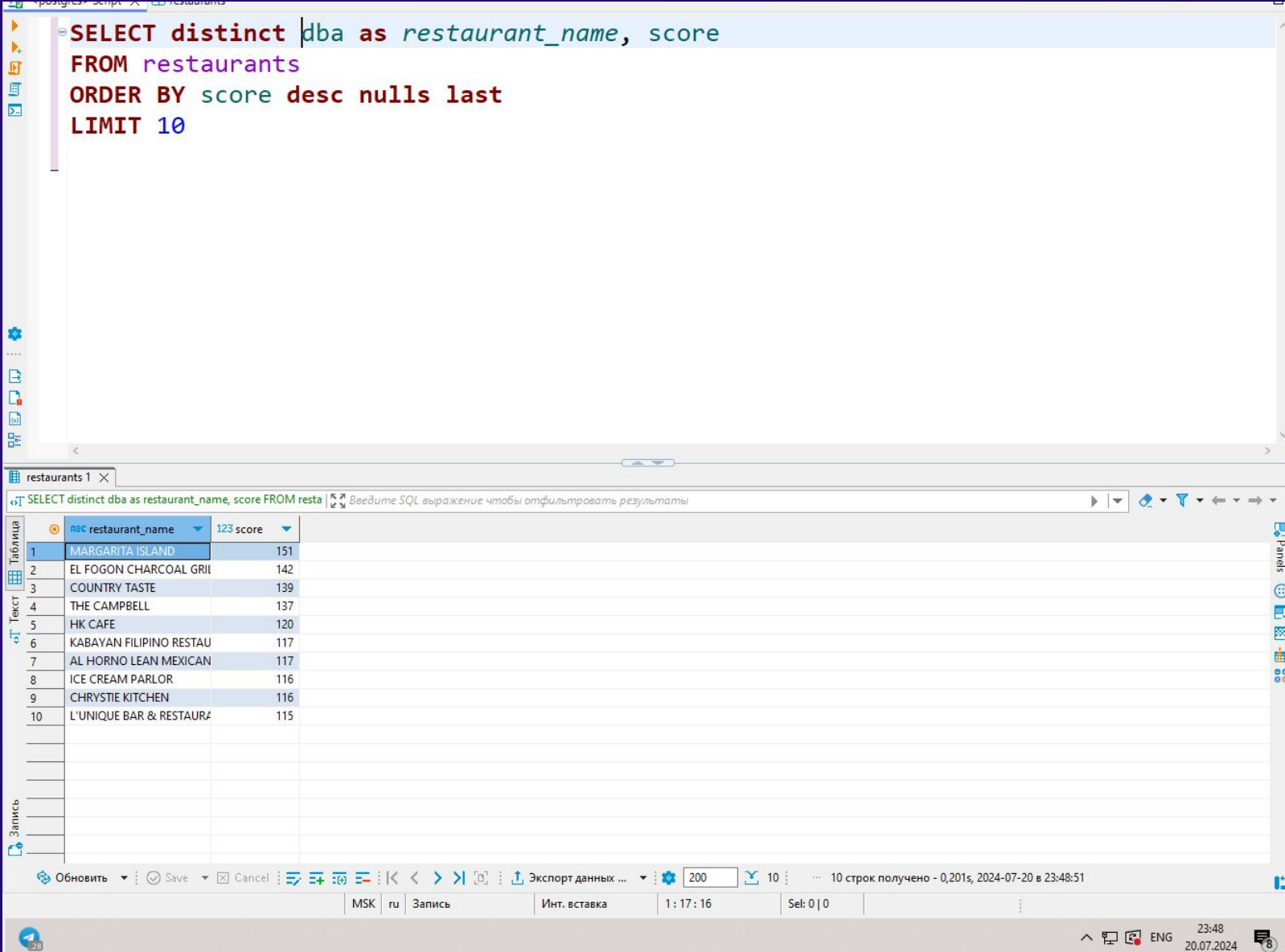
The results pane displays the output of the query, showing a table with two columns: year and inspection_count. The results are sorted by inspection_count in descending order.

year	inspection_count
2015	116 946
2016	116 053
2014	80 984
2017	79 079
2013	5 710
2012	9
2011	2

A small tooltip is visible in the bottom right corner, indicating that a screenshot fragment has been saved to the clipboard.

SQL- запрос

- Запрос 3: Выведите топ 10 самых высокооцененных ресторанов по оценке



The screenshot shows a SQL query editor with the following query:

```
SELECT distinct dba as restaurant_name, score
FROM restaurants
ORDER BY score desc nulls last
LIMIT 10
```

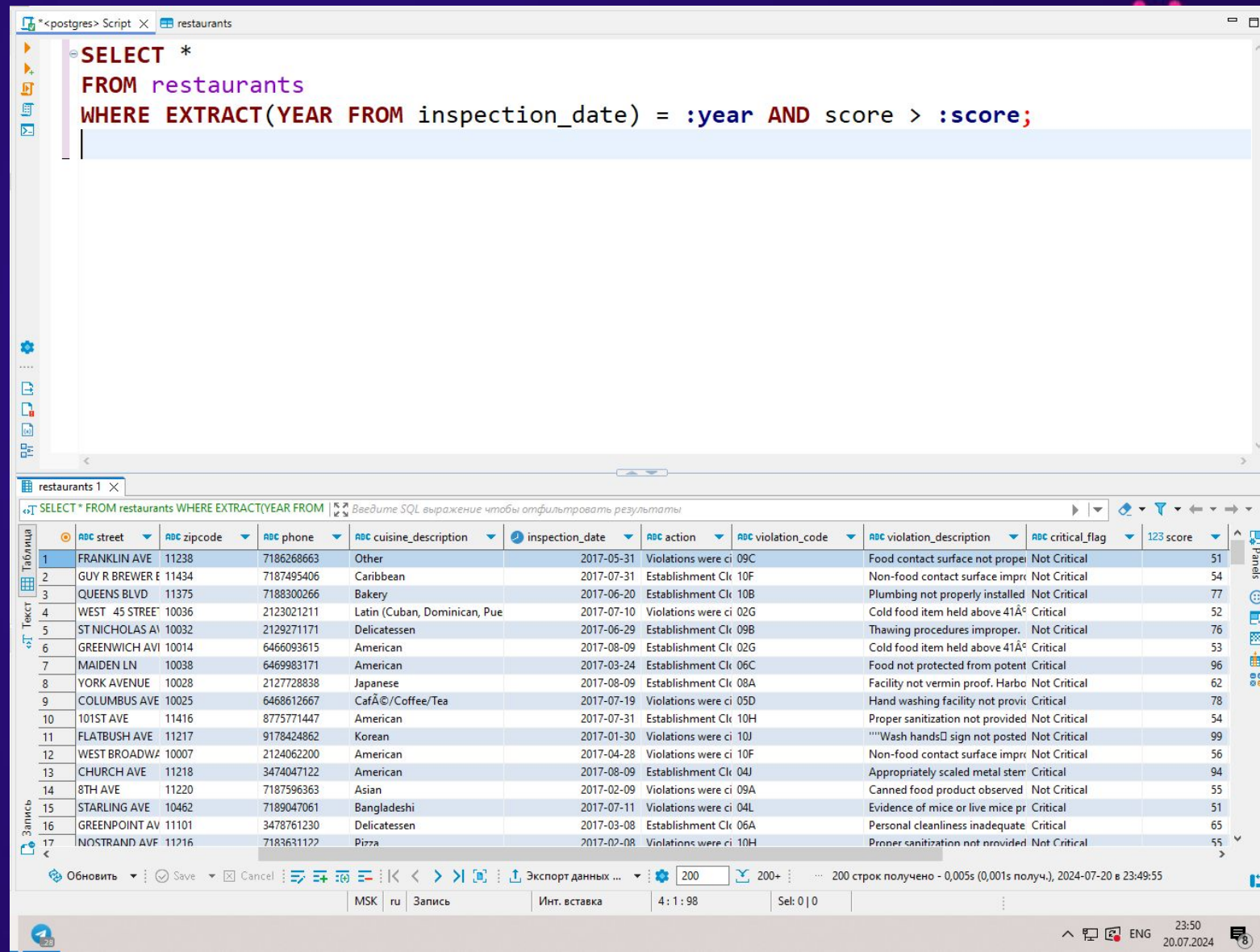
Below the query editor, the results are displayed in a table with 10 rows and 2 columns: restaurant_name and score. The table is titled "restaurants 1".

	restaurant_name	score
1	MARGARITA ISLAND	151
2	EL FOGON CHARCOAL GRILL	142
3	COUNTRY TASTE	139
4	THE CAMPBELL	137
5	HK CAFE	120
6	KABAYAN FILIPINO RESTAURANT	117
7	AL HORNO LEAN MEXICAN	117
8	ICE CREAM PARLOR	116
9	CHRYSTIE KITCHEN	116
10	L'UNIQUE BAR & RESTAURANT	115

The bottom status bar indicates that 10 rows were received, and the query was executed on 20.07.2024 at 23:48:51.

SQL- запрос

- Запрос 4: Выведите информацию о ресторанах, инспектированных в определенном году и имеющих оценку выше заданного значения (пользователь вводит год и оценку).



The screenshot displays a PostgreSQL SQL editor window with a query and its results. The query is as follows:

```
SELECT *  
FROM restaurants  
WHERE EXTRACT(YEAR FROM inspection_date) = :year AND score > :score;
```

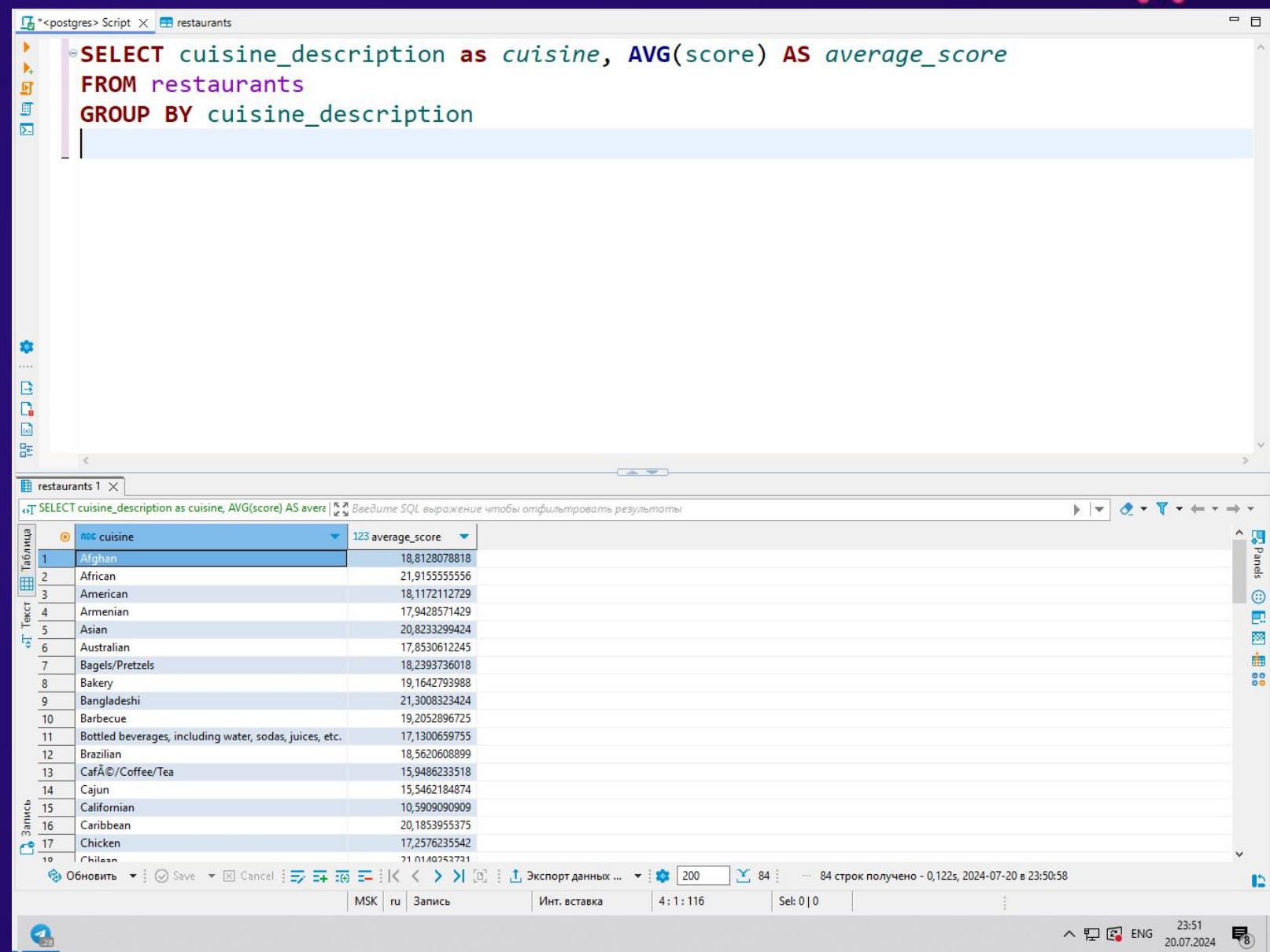
The results are shown in a table with the following columns: **ABC street**, **ABC zipcode**, **ABC phone**, **ABC cuisine_description**, **inspection_date**, **ABC action**, **ABC violation_code**, **ABC violation_description**, **ABC critical_flag**, and **123 score**. The table contains 17 rows of data, representing various restaurants and their inspection records.

	ABC street	ABC zipcode	ABC phone	ABC cuisine_description	inspection_date	ABC action	ABC violation_code	ABC violation_description	ABC critical_flag	123 score
1	FRANKLIN AVE	11238	7186268663	Other	2017-05-31	Violations were ci	09C	Food contact surface not proper	Not Critical	51
2	GUY R BREWER E	11434	7187495406	Caribbean	2017-07-31	Establishment Clk	10F	Non-food contact surface impr	Not Critical	54
3	QUEENS BLVD	11375	7188300266	Bakery	2017-06-20	Establishment Clk	10B	Plumbing not properly installed	Not Critical	77
4	WEST 45 STREET	10036	2123021211	Latin (Cuban, Dominican, Pue	2017-07-10	Violations were ci	02G	Cold food item held above 41Â°	Critical	52
5	ST NICHOLAS A	10032	2129271171	Delicatessen	2017-06-29	Establishment Clk	09B	Thawing procedures improper.	Not Critical	76
6	GREENWICH AVI	10014	6466093615	American	2017-08-09	Establishment Clk	02G	Cold food item held above 41Â°	Critical	53
7	MAIDEN LN	10038	6469983171	American	2017-03-24	Establishment Clk	06C	Food not protected from potent	Critical	96
8	YORK AVENUE	10028	2127728838	Japanese	2017-08-09	Establishment Clk	08A	Facility not vermin proof. Harbo	Not Critical	62
9	COLUMBUS AVE	10025	6468612667	CafÃ©/Coffee/Tea	2017-07-19	Violations were ci	05D	Hand washing facility not provi	Critical	78
10	101ST AVE	11416	8775771447	American	2017-07-31	Establishment Clk	10H	Proper sanitization not provided	Not Critical	54
11	FLATBUSH AVE	11217	9178424862	Korean	2017-01-30	Violations were ci	10J	""Wash hands sign not posted	Not Critical	99
12	WEST BROADWA	10007	2124062200	American	2017-04-28	Violations were ci	10F	Non-food contact surface impr	Not Critical	56
13	CHURCH AVE	11218	3474047122	American	2017-08-09	Establishment Clk	04J	Appropriately scaled metal stem	Critical	94
14	8TH AVE	11220	7187596363	Asian	2017-02-09	Violations were ci	09A	Canned food product observed	Not Critical	55
15	STARLING AVE	10462	7189047061	Bangladeshi	2017-07-11	Violations were ci	04L	Evidence of mice or live mice pr	Critical	51
16	GREENPOINT AV	11101	3478761230	Delicatessen	2017-03-08	Establishment Clk	06A	Personal cleanliness inadequate	Critical	65
17	NOSTRAND AVE	11216	7183631122	Pizza	2017-02-08	Violations were ci	10H	Proper sanitization not provided	Not Critical	55

The bottom status bar indicates: 200+ rows, 200 rows displayed, 200 rows received - 0,005s (0,001s received), 2024-07-20 at 23:49:55.

SQL- запрос

- Запрос 5: Выведите среднюю оценку ресторанов по типам кухни.



The screenshot shows a PostgreSQL SQL client interface. The top pane displays the following SQL query:

```
SELECT cuisine_description as cuisine, AVG(score) AS average_score
FROM restaurants
GROUP BY cuisine_description
```

The bottom pane shows the results of the query in a table format. The table has two columns: 'cuisine' and 'average_score'. The results are as follows:

cuisine	average_score
Afghan	18,8128078818
African	21,9155555556
American	18,1172112729
Armenian	17,9428571429
Asian	20,8233299424
Australian	17,8530612245
Bagels/Pretzels	18,2393736018
Bakery	19,1642793988
Bangladeshi	21,3008323424
Barbecue	19,2052896725
Bottled beverages, including water, sodas, juices, etc.	17,1300659755
Brazilian	18,5620608899
Café/Coffee/Tea	15,9486233518
Cajun	15,5462184874
Californian	10,5909090909
Caribbean	20,1853955375
Chicken	17,2576235542
Chilean	21,0140253731

The interface also shows a status bar at the bottom indicating that 84 rows were retrieved.

SQL- запрос

- Запрос 6: Выведите список ресторанов, расположенных в определенном районе (пользователь вводит район).

The screenshot displays a PostgreSQL database interface. At the top, a script editor window titled "restaurants" contains the following SQL query:

```
SELECT *  
FROM restaurants  
WHERE boro = :borough
```

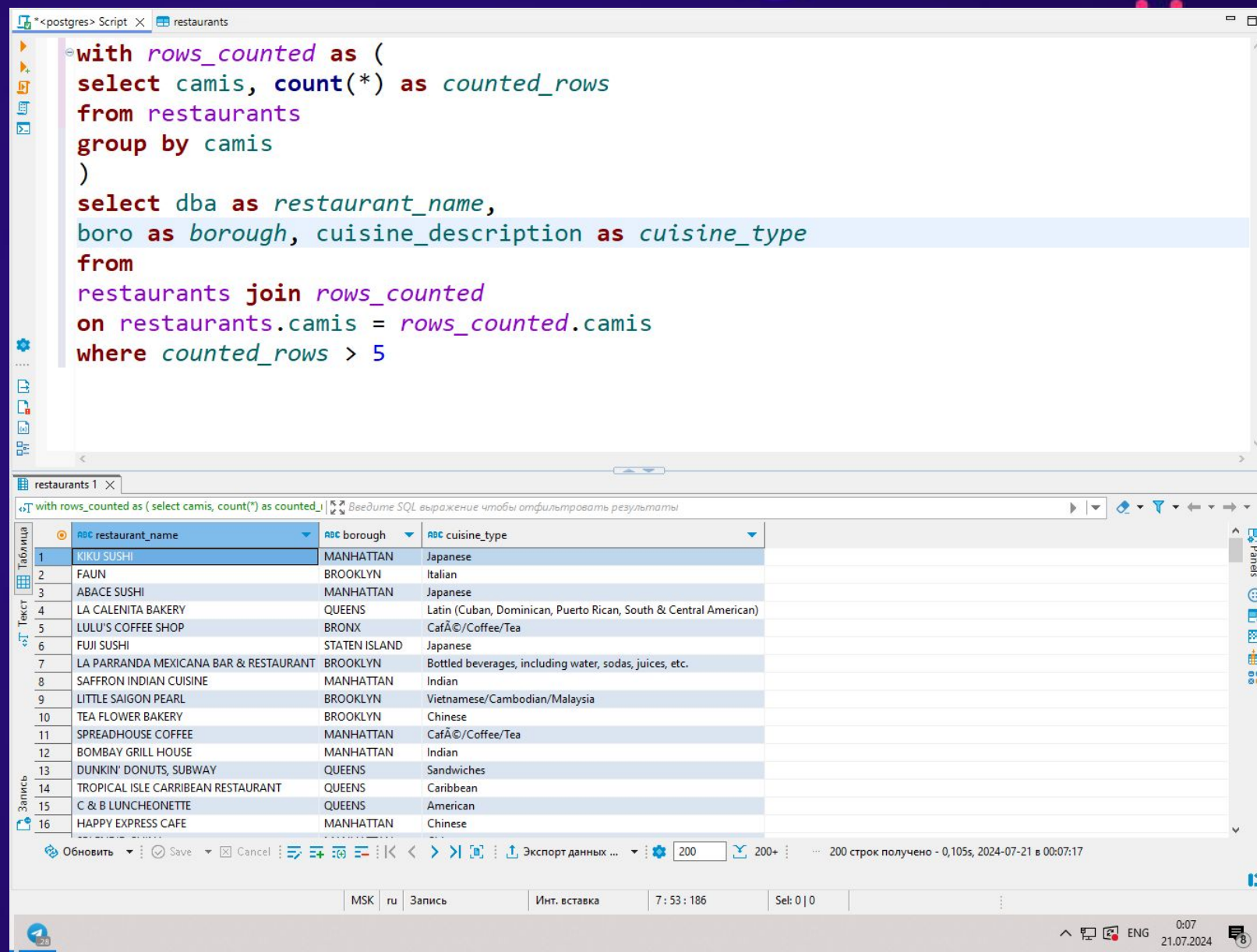
Below the script editor, a table titled "restaurants 1" shows the results of the query. The table has 17 columns: abc_camis, abc_dba, abc_boro, abc_building, abc_street, abc_zipcode, abc_phone, abc_cuisine_description, inspection_date, abc_action, and abc_violation_code. The results are displayed in a grid format with 17 rows of data.

	abc_camis	abc_dba	abc_boro	abc_building	abc_street	abc_zipcode	abc_phone	abc_cuisine_description	inspection_date	abc_action	abc_violation_code
1	50034873	DARO'S FAMO	BRONX	1752	LAFAYETTE AVE	10473	7188611333	Pizza/Italian	2015-10-01	Violations were ci	10F
2	50050795	I LOVE NY PIZZ	BRONX	2086	ARTHUR AVE	10457	3475771844	Pizza	2016-09-13	Establishment Clk	04L
3	50005591	CHINA MIA EX	BRONX	2232	WHITE PLAINS R	10467	7187086300	Chinese	2015-01-13	Violations were ci	10B
4	41250498	NO. 1 CHINESE	BRONX	7	WEST TREMONT	10453	7182945294	Chinese	2016-01-22	Violations were ci	04L
5	40861141	CASA PROMES	BRONX	308	EAST 175 STREE	10457	7189607658	American	2016-09-01	Violations were ci	08A
6	50054483	BURGER KING	BRONX	4275	WHITE PLAINS R	10466	7185146941	American	2017-06-19	Violations were ci	10D
7	41646525	CEETAY	BRONX	129	ALEXANDER AVE	10454	7186187020	Japanese	2014-03-14	Violations were ci	02B
8	41533970	DUNKIN' DONI	BRONX	2370	GRAND CONCO	10458	7182204828	American	2017-06-13	Violations were ci	02G
9	41077507	LA NUEVA ESTI	BRONX	390	EAST 204 STREE	10467	7186534111	Latin (Cuban, Dominican, Pue	2014-08-27	Violations were ci	06D
10	41380595	EL PRESIDENTE	BRONX	4-10	EAST 208 STREE	10467	7186555245	Latin (Cuban, Dominican, Pue	2016-11-23	Violations were ci	10I
11	50013618	PATSY'S PIZZE	BRONX	980	MORRIS PARK A	10462	7186762527	Pizza/Italian	2014-09-20	Violations were ci	16B
12	50044525	UNCLE AL'S	BRONX	3841	E TREMONT AVE	10465	9178076639	Soul Food	2017-04-03	Violations were ci	06D
13	41612730	PAPA YE RESTA	BRONX	196	MCCELLELLAN ST	10456	7186813240	African	2014-02-10	Violations were ci	08A
14	41286021	DUNKIN' DONI	BRONX	2241	SOUTHERN BOU	10460	7182204019	Donuts	2014-10-08	Violations were ci	10F
15	41077507	LA NUEVA ESTI	BRONX	390	EAST 204 STREE	10467	7186534111	Latin (Cuban, Dominican, Pue	2016-06-05	Violations were ci	02B
16	41413315	GEORGE'S DINI	BRONX	2369	WESTCHESTER A	10462	7188281170	American	2015-12-07	Violations were ci	06E
17	41370989	CESTRA'S PIZZ	BRONX	3617	EAST TREMONT.	10465	7185187900	Pizza	2016-12-29	Violations were ci	04L

The interface also shows a status bar at the bottom indicating "200 строк получено - 0,002s (0,001s получ.)", "2024-07-20 в 23:52:35", and "MSK ru Запись".

SQL- запрос

- Запрос 7: Выведите информацию о ресторанах, имеющих определенное количество нарушений (например, более 5 нарушений).



The screenshot shows a PostgreSQL script editor with a SQL query and its results in a table view.

```
with rows_counted as (  
  select camis, count(*) as counted_rows  
  from restaurants  
  group by camis  
)  
select dba as restaurant_name,  
boro as borough, cuisine_description as cuisine_type  
from  
restaurants join rows_counted  
on restaurants.camis = rows_counted.camis  
where counted_rows > 5
```

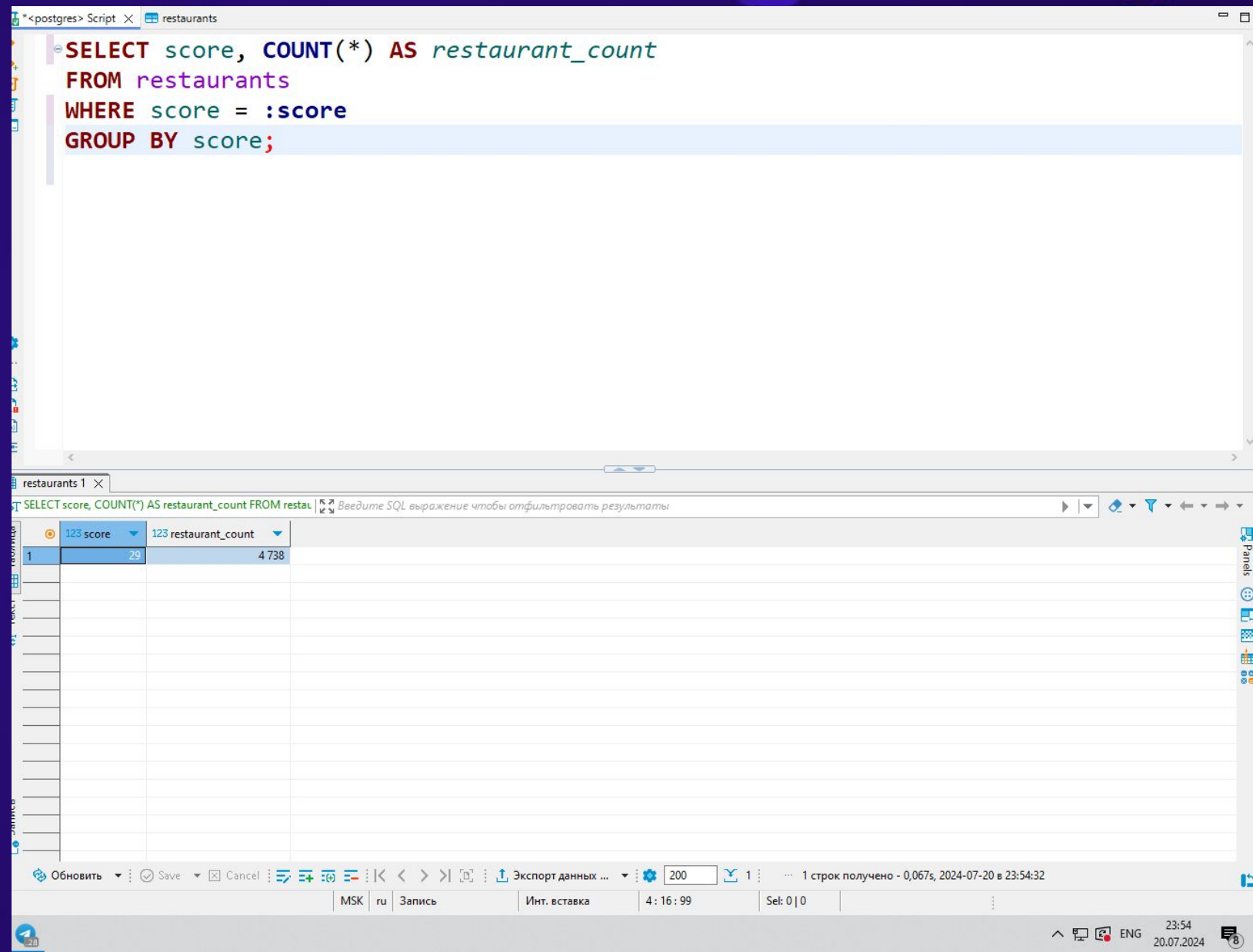
The results table, titled "restaurants 1", displays the following data:

	restaurant_name	borough	cuisine_type
1	KIKU SUSHI	MANHATTAN	Japanese
2	FAUN	BROOKLYN	Italian
3	ABACE SUSHI	MANHATTAN	Japanese
4	LA CALENITA BAKERY	QUEENS	Latin (Cuban, Dominican, Puerto Rican, South & Central American)
5	LULU'S COFFEE SHOP	BRONX	Café/Coffee/Tea
6	FUJI SUSHI	STATEN ISLAND	Japanese
7	LA PARRANDA MEXICANA BAR & RESTAURANT	BROOKLYN	Bottled beverages, including water, sodas, juices, etc.
8	SAFFRON INDIAN CUISINE	MANHATTAN	Indian
9	LITTLE SAIGON PEARL	BROOKLYN	Vietnamese/Cambodian/Malaysia
10	TEA FLOWER BAKERY	BROOKLYN	Chinese
11	SPREADHOUSE COFFEE	MANHATTAN	Café/Coffee/Tea
12	BOMBAY GRILL HOUSE	MANHATTAN	Indian
13	DUNKIN' DONUTS, SUBWAY	QUEENS	Sandwiches
14	TROPICAL ISLE CARRIBEAN RESTAURANT	QUEENS	Caribbean
15	C & B LUNCHEONETTE	QUEENS	American
16	HAPPY EXPRESS CAFE	MANHATTAN	Chinese

The interface includes a sidebar with icons for various database operations, a status bar at the bottom showing "200 строк получено" (200 rows received), and a system tray with the date and time.

SQL- запрос

- Запрос 8: Выведите количество ресторанов с определенной оценкой.



The screenshot displays a PostgreSQL client interface with a script editor and a results pane. The script editor contains the following SQL query:

```
SELECT score, COUNT(*) AS restaurant_count  
FROM restaurants  
WHERE score = :score  
GROUP BY score;
```

The results pane shows a table with two columns: 'score' and 'restaurant_count'. The first row of data shows a score of 29 and a count of 4738.

score	restaurant_count
29	4738

The bottom status bar indicates that 1 row was received in 0.067s on 2024-07-20 at 23:54:32.

SQL- запрос

- Запрос 9: Выведите список ресторанов, прошедших инспекцию с наилучшим результатом.

The screenshot shows a PostgreSQL script editor window with a script named 'restaurants'. The script contains the following SQL query:

```
SELECT * FROM restaurants WHERE score = (SELECT MAX(score) FROM restaurants);
```

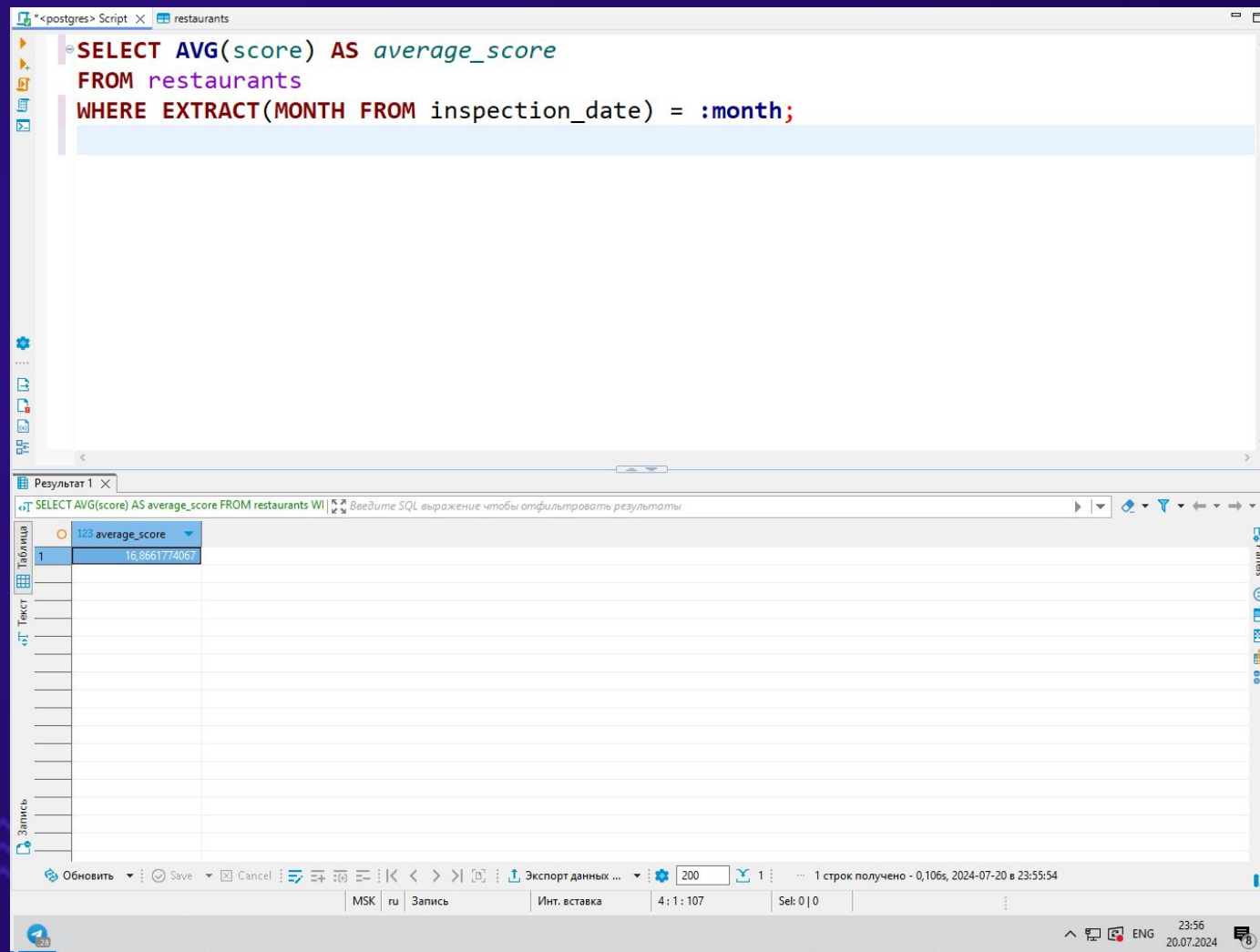
Below the script editor, the query results are displayed in a table. The table has 10 columns: 'camis', 'dba', 'boro', 'building', 'street', 'zipcode', 'phone', 'cuisine_description', 'inspection_date', and 'action'. The results show 10 rows of data, all with a score of 100. The 'action' column for all rows is 'Establishment Closures'.

	camis	dba	boro	building	street	zipcode	phone	cuisine_description	inspection_date	action
1	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
2	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
3	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
4	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
5	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
6	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
7	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
8	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
9	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures
10	50050373	MARGARITA ISLAND	BROOKLYN	1105	BOWERY ST	11224	7184491284	Bottled beverages, including water, sodas, juices, etc.	2017-07-20	Establishment Closures

The bottom of the screenshot shows the status bar with the following information: 'Обновить', 'Save', 'Cancel', 'Экспорт данных ...', '200', '10', '10 строк получено - 0,140с (0,001с получ.), 2024-07-20 в 23:55:23', 'MSK', 'ru', 'Запись', 'Инт. вставка', '4:1:81', 'Set: 0 | 0', '23:55', '20.07.2024', 'ENG', and a system icon.

SQL- запрос

- Запрос 10: Выведите среднюю оценку ресторанов, инспектированных в определенном месяце (пользователь вводит месяц как число).



The screenshot shows a PostgreSQL client window with the following SQL query:

```
SELECT AVG(score) AS average_score
FROM restaurants
WHERE EXTRACT(MONTH FROM inspection_date) = :month;
```

The result is displayed in a table with one row:

1	average_score
1	16.8661774067

The interface also shows a status bar at the bottom indicating that 1 row was received on 2024-07-20 at 23:55:54.

Триггеры: таблица для логов

```
CREATE TABLE log (  
  id SERIAL PRIMARY KEY,  
  log_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  operation TEXT,  
  details TEXT  
);
```

Триггер

- Триггер 1: При добавлении нового ресторана в базу данных, автоматически записывать в лог информацию о дате и времени добавления

The screenshot displays a PostgreSQL database management interface. At the top, there are tabs for 'restaurants', 'log', and '*<postgres> Console'. The 'log' tab is active, showing a table with columns: 'id', 'log_time', 'operation', and 'details'. The table contains one row with the following data: id=5, log_time=2024-07-21 01:36:54.323, operation=INSERT, and details='New restaurant added with CAMIS: 123456'.

Below the table, the SQL console shows the following code:

```
CREATE OR REPLACE FUNCTION log_insert()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log (operation, details)
    VALUES ('INSERT', 'New restaurant added with CAMIS: ' || NEW.camis);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_insert
AFTER INSERT ON restaurants
FOR EACH ROW
EXECUTE FUNCTION log_insert();
```

Триггер

- Триггер 2: При изменении оценки ресторана, автоматически обновлять его описание в базе

```
CREATE OR REPLACE FUNCTION update_description()
RETURNS TRIGGER AS $$
BEGIN
    NEW.violation_description := 'Updated score to ' || NEW.score;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_description
BEFORE UPDATE OF score ON restaurants
FOR EACH ROW
EXECUTE FUNCTION update_description();
```

```
-- Шаг 1: Создание начальной записи
-- Изначальный score = 85
-- Описание: Initial violation description
INSERT INTO restaurants (camis, dba, boro, building, street, zipcode, phone, cuisine_description, inspection_date, action_date)
VALUES ('123457', 'Test Restaurant', 'Brooklyn', '456', 'Another St', '11201', '0987654321', 'Chinese', '2023-07-20', 'A');
```

```
-- Шаг 2: Обновление оценки ресторана
UPDATE restaurants SET score = 42 WHERE camis = '123457';
```

```
SELECT violation_description FROM restaurants WHERE camis = '123457';
```

```
SELECT violation_description FROM restaurants
```

	ABC violation_description	
1	Updated score to 42	
2	Updated score to 42	

Триггер

- Триггер 3: При удалении ресторана из базы данных, записывать в лог информацию о дате и времени удаления

log Введите SQL выражение чтобы отфильтровать результаты					
Таблица	id	log_time	operation	details	
1	5	2024-07-21 01:36:54.323	INSERT	New restaurant added with CAMIS: 123456	
2	6	2024-07-21 01:45:14.175	DELETE	Restaurant deleted with CAMIS: 123456	

```
CREATE OR REPLACE FUNCTION log_delete()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log (operation, details)
    VALUES ('DELETE', 'Restaurant deleted with CAMIS: ' || OLD.camis);
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_delete
AFTER DELETE ON restaurants
FOR EACH ROW
EXECUTE FUNCTION log_delete();
```

Триггер

- Триггер 4: При добавлении нового типа кухни, автоматически создавать таблицу для хранения информации о ресторанах этого типа

```
CREATE OR REPLACE FUNCTION create_cuisine_table()
RETURNS TRIGGER AS $$
BEGIN
    EXECUTE 'CREATE TABLE IF NOT EXISTS ' || replace(NEW.cuisine_description, ' ', '_') ||
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_create_cuisine_table
AFTER INSERT ON restaurants
FOR EACH ROW
WHEN (NEW.cuisine_description IS DISTINCT FROM NULL)
EXECUTE FUNCTION create_cuisine_table();
```

The screenshot shows a PostgreSQL database interface. On the left, a tree view displays the database structure: postgres - localhost:5432, Базы данных, postgres, Схемы, public, restaurants, and Таблицы. The tables listed are french_restaurants (8K), log (32K), restaurants (151M), and teremok_restaurants (8K). On the right, the SQL console shows a comment "-- Вставка нового ресторана" followed by an SQL insert statement: INSERT INTO restaurants (camis, dba, boro, building, street, zipcode, phone, cuisine_description, inspection_date, action_date) VALUES ('4321', 'Sample Restaurant', 'Manhattan', '123', 'Main St', '10001', '1234567890', 'Teremok', '2023-07-20', 'Act').

Триггер

- Триггер 5: При изменении информации о районе, автоматически обновлять информацию о

```
CREATE OR REPLACE FUNCTION update_boro_log()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log (operation, details)
    VALUES ('UPDATE', 'Borough updated to: ' || NEW.boro || ' for CAMIS: ' || OLD.camis);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trigger_update_boro
AFTER UPDATE OF boro ON restaurants
FOR EACH ROW
EXECUTE FUNCTION update_boro_log();
```

⊖ -- Шаг 1: Создание начальной записи

```
INSERT INTO restaurants (camis, dba, boro, building, street, zipcode, phone, cuisine_description, inspection_date, active)
VALUES ('123458', 'Another Test Restaurant', 'Queens', '789', 'Third St', '11368', '1122334455', 'Mexican', '2023-07-20', 1);
```

⊖ -- Шаг 2: Обновление района ресторана

```
UPDATE restaurants SET boro = 'Manhattan' WHERE camis = '123458';
```


Процедура

- Процедура 1:
Возвращать список ресторанов, соответствующих заданным критериям поиска (тип кухни, район, дата инспекции, оценка).

```
CREATE OR REPLACE FUNCTION search_restaurants(  
    search_cuisine TEXT,  
    search_boro TEXT,  
    search_inspection_date DATE,  
    search_score INT  
)  
RETURNS TABLE (  
    camis INTEGER,  
    dba TEXT,  
    cuisine_description TEXT,  
    inspection_date DATE,  
    score INTEGER  
) AS $$  
BEGIN  
    RETURN QUERY  
    SELECT *  
    FROM restaurants  
    WHERE  
        (search_cuisine IS NULL OR cuisine_description = search_cuisine) AND  
        (search_inspection_date IS NULL OR inspection_date = search_inspection_date) AND  
        (search_score IS NULL OR score >= search_score);  
END;  
$$ LANGUAGE plpgsql;
```


Процедура

- Процедура 2:
Вычислять среднюю
оценку ресторанов
определенного типа
кухни.

```
CREATE OR REPLACE FUNCTION average_score_by_cuisine(cuisine TEXT)
RETURNS FLOAT AS $$
DECLARE
    avg_score FLOAT;
BEGIN
    SELECT AVG(score)
    INTO avg_score
    FROM restaurants
    WHERE cuisine_description = cuisine;

    RETURN avg_score;
END;
$$ LANGUAGE plpgsql;
```

restaurants	log	*<postgres> Console X
SELECT average_score_by_cuisine('Chinese');		
1	123 average_score_by_cuisine	20,6038843056

Процедура

- Процедура 3:
Определять топ 5 самых высокооцененных ресторанов по оценке.

```
CREATE OR REPLACE FUNCTION top_5_restaurants()  
RETURNS TABLE (  
    camis INTEGER,  
    dba TEXT,  
    boro TEXT,  
    building TEXT,  
    street TEXT,  
    zipcode TEXT,  
    phone TEXT,  
    cuisine_description TEXT,  
    inspection_date DATE,  
    action TEXT,  
    violation_code TEXT,  
    violation_description TEXT,  
    critical_flag TEXT,  
    score INTEGER,  
    grade TEXT,  
    grade_date DATE,  
    record_date DATE,  
    inspection_type TEXT  
) AS $$  
BEGIN  
    RETURN QUERY  
    SELECT camis, dba, max(score)  
    FROM restaurants  
    group by camis  
    ORDER BY score DESC  
    LIMIT 5;  
END;  
$$ LANGUAGE plpgsql;
```

Процедура

- Процедура 4:
Выводить статистику по ресторанам (количество ресторанов по типам кухни, районам, датам инспекции) в виде таблицы.

```
CREATE OR REPLACE FUNCTION restaurant_statistics()  
RETURNS TABLE (  
    cuisine_description TEXT,  
    boro TEXT,  
    inspection_date DATE,  
    restaurant_count INT  
) AS $$  
BEGIN  
    RETURN QUERY  
    SELECT  
        cuisine_description,  
        boro,  
        inspection_date,  
        COUNT(*) as restaurant_count  
    FROM restaurants  
    GROUP BY cuisine_description, boro, inspection_date  
    ORDER BY cuisine_description, boro, inspection_date;  
END;  
$$ LANGUAGE plpgsql;
```


Процедура

- Процедура 5:
Вычислять среднее
количество нарушений
ресторанов,
инспектированных в
определенном году.

```
CREATE OR REPLACE FUNCTION average_violations_by_year(inspect_year INT)
RETURNS FLOAT AS $$
DECLARE
    avg_violations FLOAT;
BEGIN
    SELECT AVG(violation_count)
    INTO avg_violations
    FROM (
        SELECT camis, COUNT(violation_code) AS violation_count
        FROM restaurants
        WHERE EXTRACT(YEAR FROM inspection_date) = inspect_year
        GROUP BY camis
    ) AS violations_per_restaurant;

    RETURN avg_violations;
END;
$$ LANGUAGE plpgsql;
```

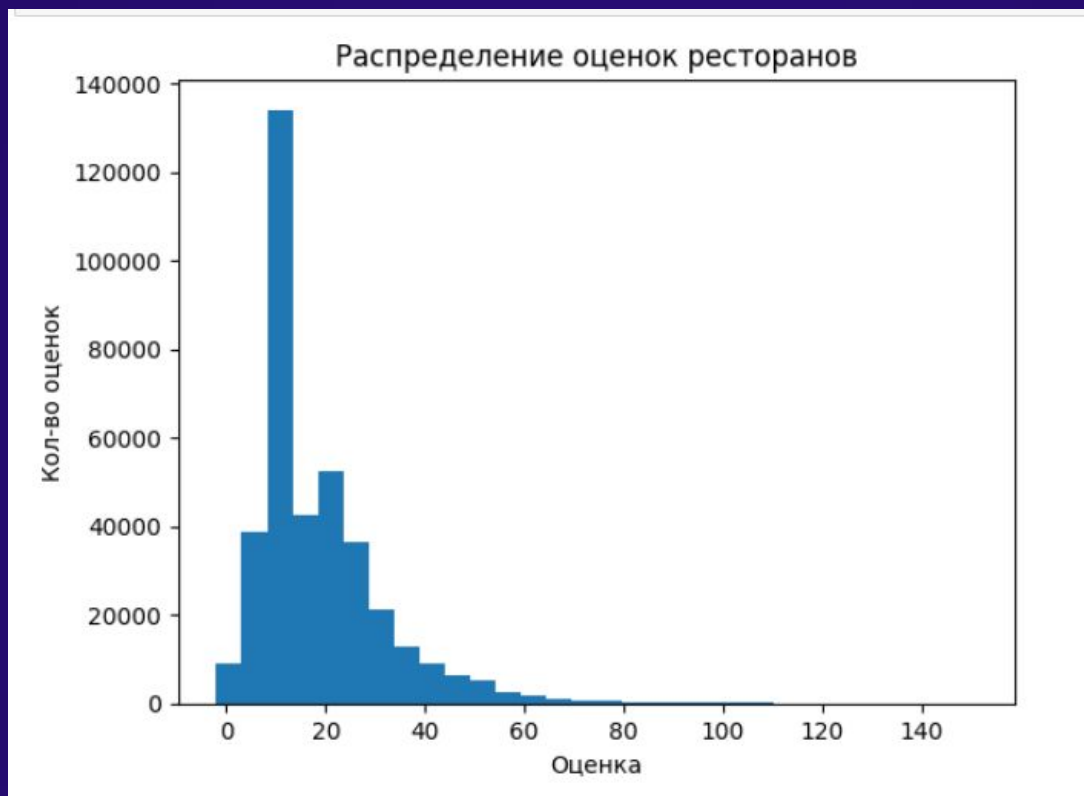
```
SELECT average_violations_by_year(2023);
```

123 average_violations_by_year	
1	1,666666667

The background is a solid dark blue. It features several abstract geometric elements: a large, semi-transparent light blue arrow pointing downwards from the top left towards the center; a series of vertical red bars of varying heights in the top left; a grid of small red and orange dots in the bottom left; a series of horizontal wavy lines in the bottom center; and a cluster of red and orange dots in the bottom right. There are also some small white and yellow plus signs scattered in the upper left area.

Визуализация данных

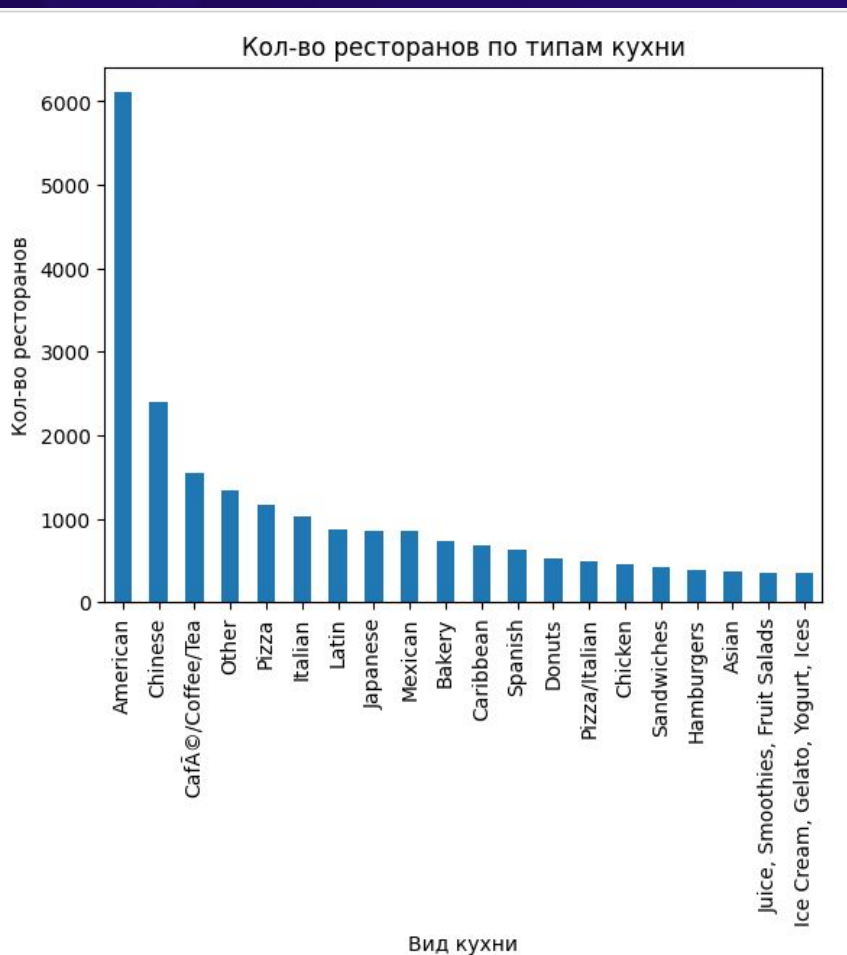
Распределение оценок ресторанов.



: # Выборка 1: Постройте гистограмму распределения оценок ресторанов.

```
# Построение гистограммы
plt.hist(df['score'], bins=30)
plt.xlabel('Оценка')
plt.ylabel('Кол-во оценок')
plt.title('Распределение оценок ресторанов')
plt.show()
```

Количество ресторанов по типам кухни.



Выборка 2: С помощью диаграммы столбцов сравните количество ресторанов по типам кухни.

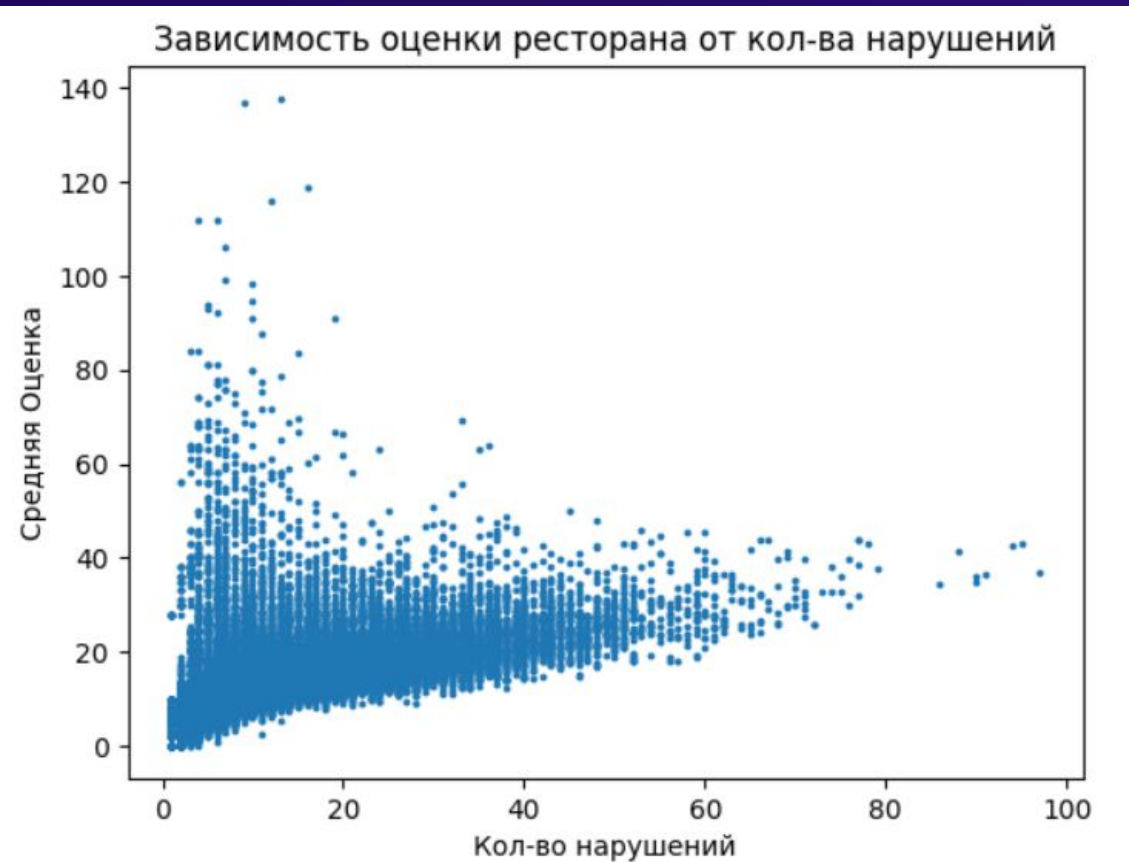
Кол-во ресторанов по кухням

```
cuisine_counts = df.groupby(['cuisine description'])['camis'].nunique()  
cuisine_counts = cuisine_counts.sort_values(ascending=False).head(20)
```

Построение диаграммы столбцов

```
cuisine_counts.plot(kind='bar')  
plt.xlabel('Вид кухни')  
plt.ylabel('Кол-во ресторанов')  
plt.title('Кол-во ресторанов по типам кухни')  
plt.show()
```

Scatter plot зависимости оценки ресторана от количества нарушений.



```
# Выборка 3: Создайте scatter plot зависимости оценки ресторана от количества нарушений.  
  
# Группировка значений  
violation_count = df.groupby(['camis']).agg({'score': 'mean', 'critical flag': 'count'})  
  
# Построение scatter plot  
plt.scatter(violation_count['critical flag'], violation_count['score'], s=3)  
plt.xlabel('Кол-во нарушений')  
plt.ylabel('Средняя Оценка')  
plt.title('Зависимость оценки ресторана от кол-ва нарушений')  
plt.show()
```


Line chart зависимости среднего рейтинга ресторанов от времени.

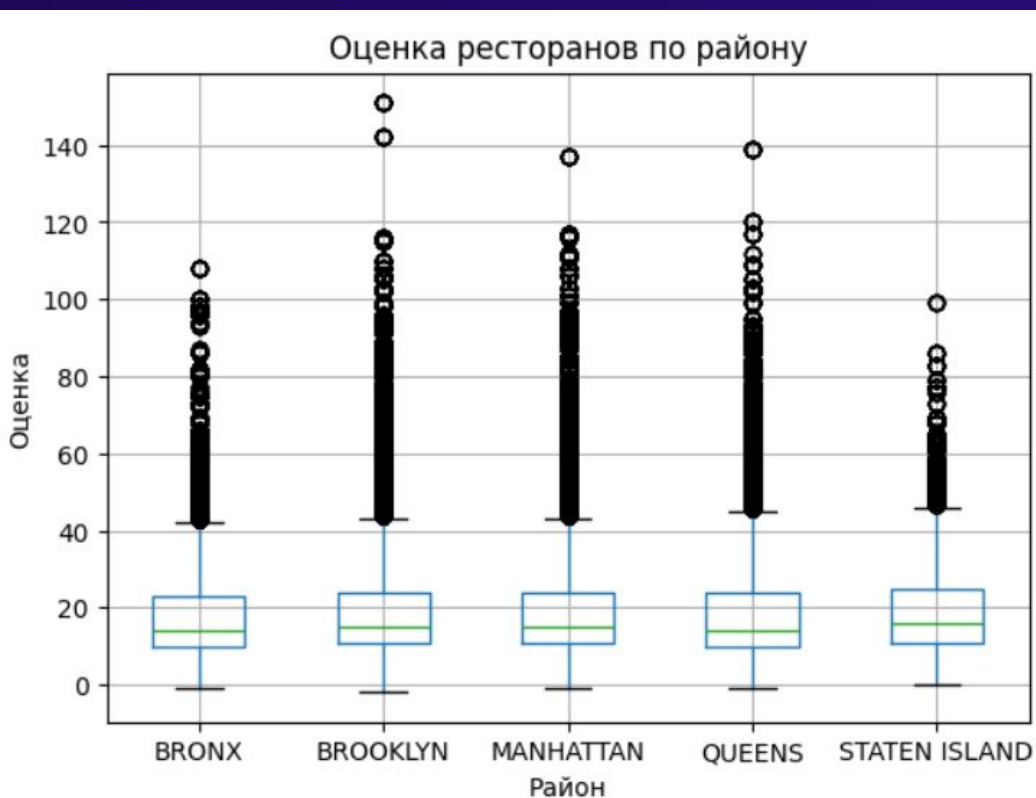


Выборка 4: Постройте line chart зависимости среднего рейтинга ресторанов от времени.

Группировка данных по году и расчет среднего рейтинга
avg_scores_by_year = df.groupby(df['inspection date'].dt.year)['score'].mean()

Построение line chart
avg_scores_by_year.plot(kind='line')
plt.xlabel('Год')
plt.ylabel('Средняя оценка')
plt.title('Средняя оценка ресторанов от времени проверок')
plt.show()

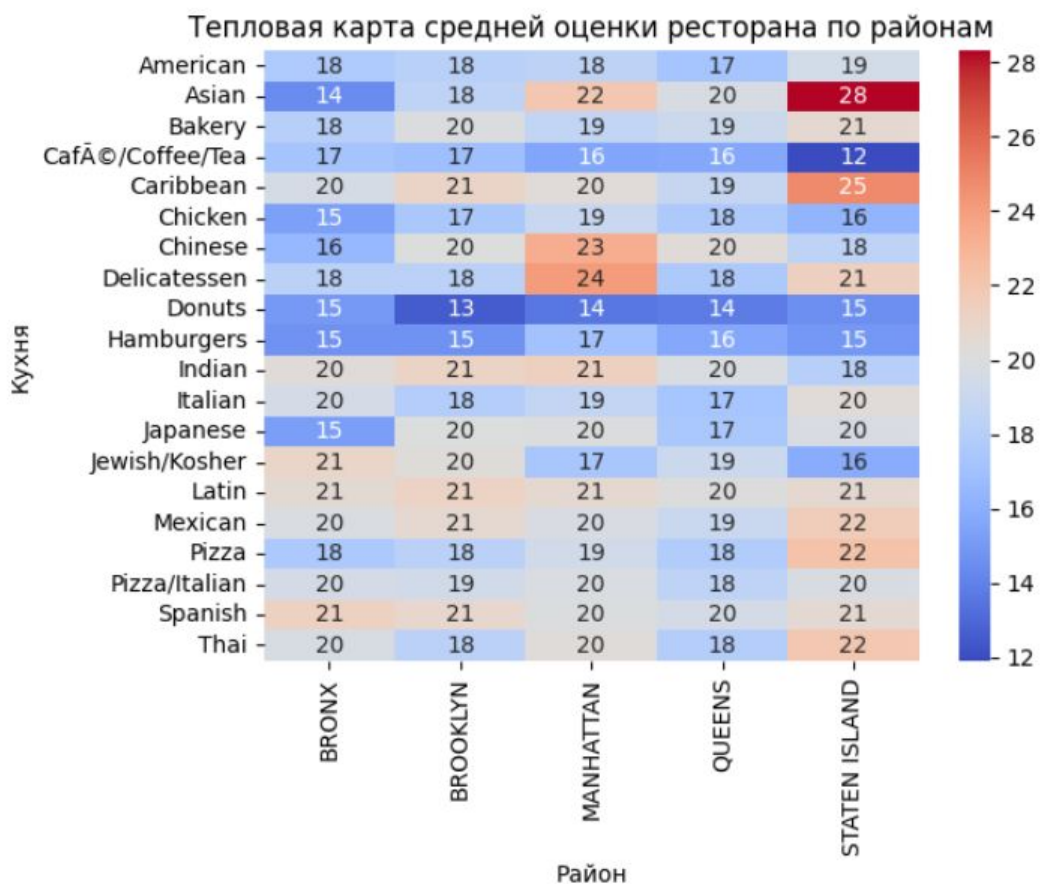
Box plot распределения оценок ресторанов, расположенных в разных районах.



Выборка 5: С помощью box plot сравните распределение оценок ресторанов, расположенных в разных районах.

```
# Построения boxplot
df.boxplot(column='score', by='boro')
plt.xlabel('Район')
plt.ylabel('Оценка')
plt.title('Оценка ресторанов по району')
plt.suptitle('')
plt.show()
```

Тепловая карта зависимости оценки ресторана от типа кухни и района.



```
# Выборка 6: Создайте тепловую карту (heatmap) зависимости оценки ресторана от типа кухни и района.

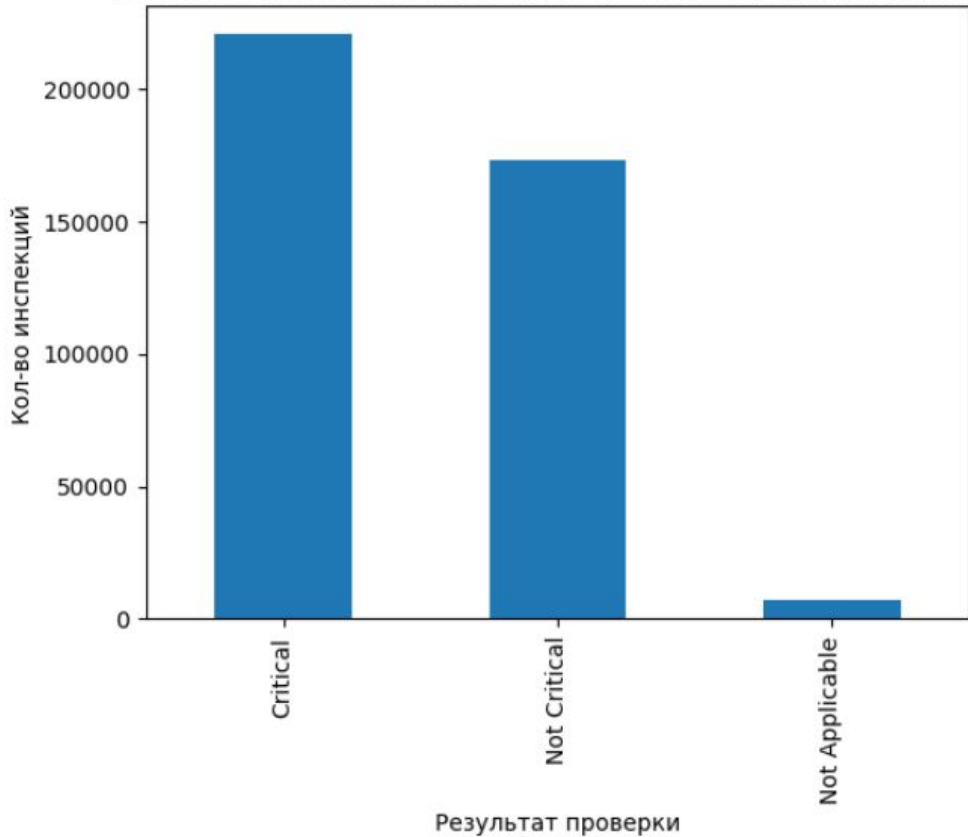
# Возьмем только топ 20 видов кухонь, так как их слишком много
cuisine_counts = df['cuisine description'].value_counts().head(20)
cuisine_top_counts = cuisine_counts.index.tolist()
temp_df = df[df['cuisine description'].isin(cuisine_top_counts)]

# Создание сводной таблицы
pivot_table = temp_df.pivot_table(values='score', index='cuisine description', columns='boro', aggfunc='mean')

# Построение тепловой карты
sns.heatmap(pivot_table, cmap='coolwarm', annot=True)
plt.xlabel('Район')
plt.ylabel('Кухня')
plt.title('Тепловая карта средней оценки ресторана по районам')
plt.show()
```

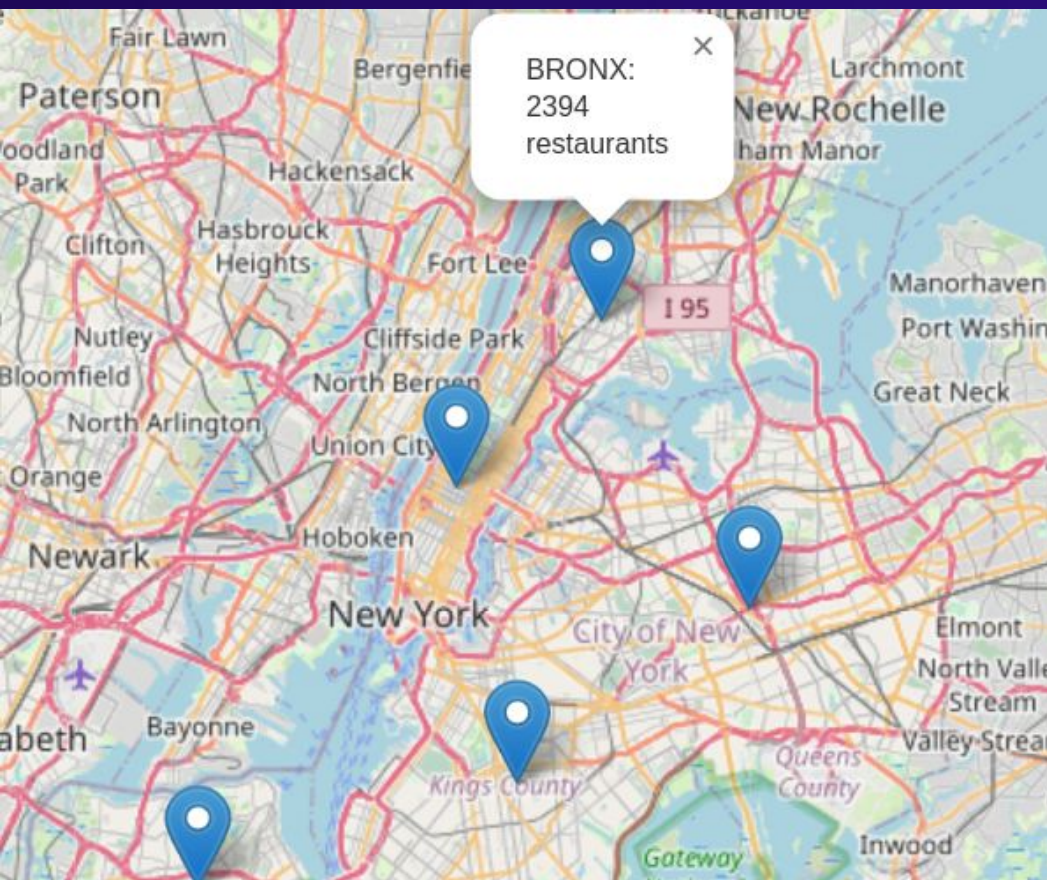
Bar chart, по количеству ресторанов, инспектированных с разными результатами

Зависимость оценки инспекции от общего числа инспекций



```
: # Выборка 7: Постройте bar chart, сравнивающий количество ресторанов, ин  
# Корректнее считать кол-во инспекций и к каким результатам они привели  
  
# Группировка данных по результатам инспекции  
result_counts = df['critical flag'].value_counts()  
# Построение bar chart  
result_counts.plot(kind='bar')  
plt.xlabel('Результат проверки')  
plt.ylabel('Кол-во инспекций')  
plt.title('Зависимость оценки инспекции от общего числа инспекций')  
plt.show()
```


Карта количества ресторанов в разных районах Нью-Йорка.



Выборка 9: Создайте карту, отображающую количество ресторанов в разных районах Нью-Йорка.

```
#Кол-во ресторанов по районам и по коду (повторяющиеся коды склеиваются)
borough_counts = df.groupby(['boro'])['camis'].nunique()
```

Создание карты

```
nyc_map = folium.Map(location=[40.7128, -74.0060], zoom_start=10)
```

#Геоданные

```
locations = {
    'BRONX': [40.82759731456553, -73.90654192980843],
    'MANHATTAN': [40.76182261353115, -73.98152554163758],
    'BROOKLYN': [40.645311141825836, -73.95045756539682],
    'QUEENS': [40.71442138576018, -73.82992316133488],
    'STATEN ISLAND': [40.603857162374375, -74.11700994930406]
}
```

Добавление маркеров на карту

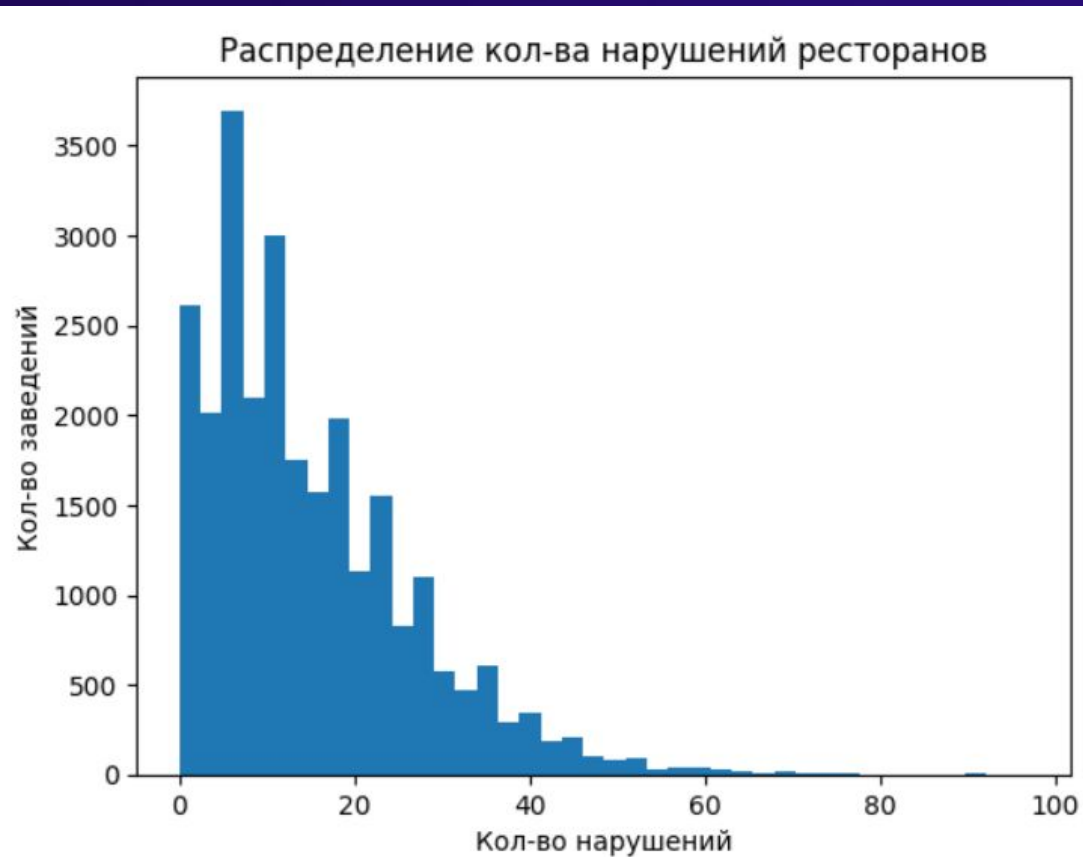
```
marker_cluster = MarkerCluster().add_to(nyc_map)
```

```
for borough, count in borough_counts.items():
    folium.Marker(location=locations[borough], popup=f'{borough}: {count} restaurants').add_to(marker_cluster)
```

Отображение карты

```
nyc_map
```

Гистограмма распределения количества нарушений ресторанов.



Выборка 10: Постройте гистограмму распределения количества нарушений ресторанов.

Группировка по ресторанам

```
violation_count = df.groupby(['camis']).agg({'dba': 'count'})
```

Построение гистограммы

```
plt.hist(violation_count['dba'], bins=40)
```

```
plt.xlabel('Кол-во нарушений')
```

```
plt.ylabel('Частота')
```

```
plt.title('Распределение кол-ва нарушений ресторанов')
```

```
plt.show()
```



Развитие проекта

Возможен более детальный анализ зависимости между кол-вом проверок и типом кухни, а также связь между оценками ресторанов в различные годы.

Также можно найти дататсет аналогичный текущему (например другой американский город), проделать ту же работу и сравнить результаты.



Контакты

Telegram: @nvladimiri

email: nikvladimir27@gmail.com