

Департамент образования города Москвы

Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

Лабораторная работа 3
по дисциплине «Проектный практикум по разработке ETL-решений»

Тема: «Практическая работа на вебинаре»

Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
St_88

Москва
2025

ЗАДАНИЕ 2.

Анализ и визуализация данных в ClickHouse

Цель: провести анализ данных, загруженных в таблицу `person_count_by_city` в ClickHouse, и создать визуализации для полученных результатов.

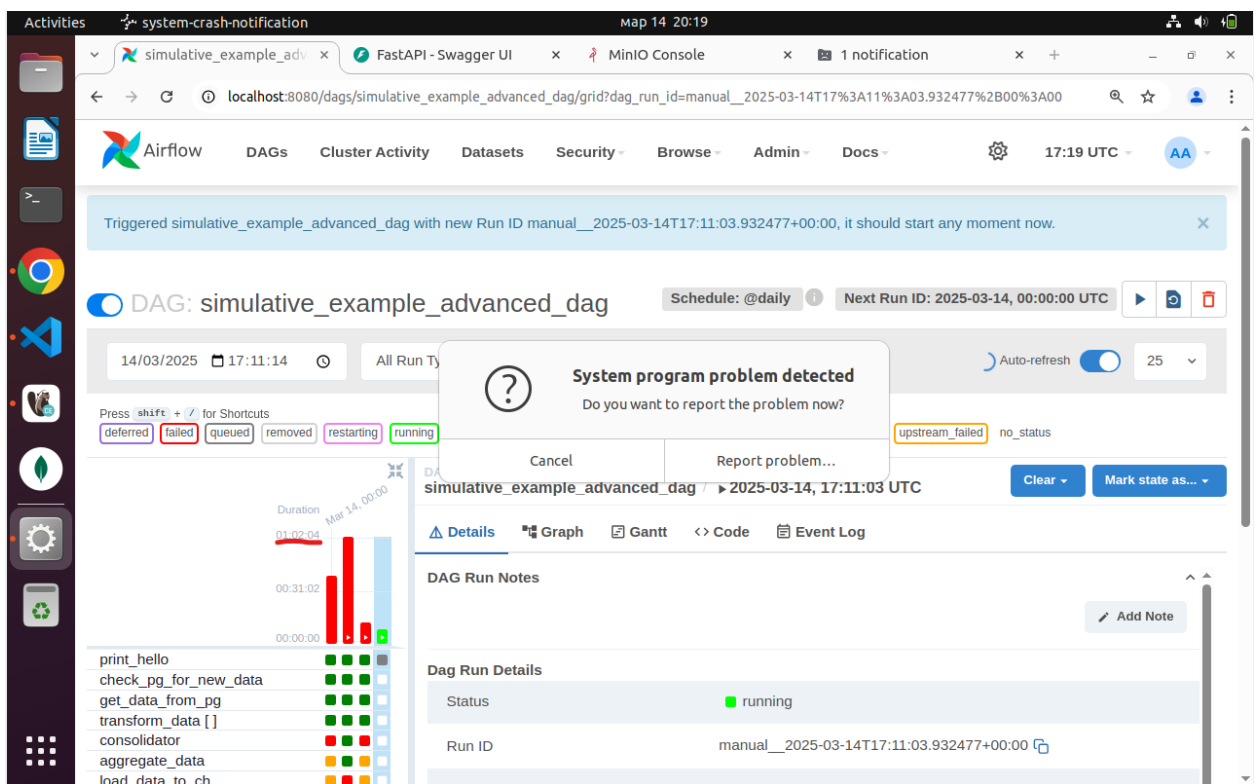
Задачи:

1. Анализ количества людей по городам.
 - a. Найдите топ-10 городов с наибольшим количеством людей.
 - b. Найдите среднее количество людей в городах.
2. Анализ распределения количества людей по городам:
 - a. Постройте гистограмму распределения количества людей по городам.
3. Анализ городов с минимальным и максимальным количеством людей:
 - a. Найдите город с минимальным количеством людей.
 - b. Найдите город с максимальным количеством людей.
4. Визуализация данных. Создайте графики для визуализации результатов анализа:
 - a. Диаграмма топ-10 городов по количеству проживающих.
 - b. Гистограмма распределения количества людей по городам.
 - c. Линейный график, показывающий минимальное и максимальное количество людей в городах.

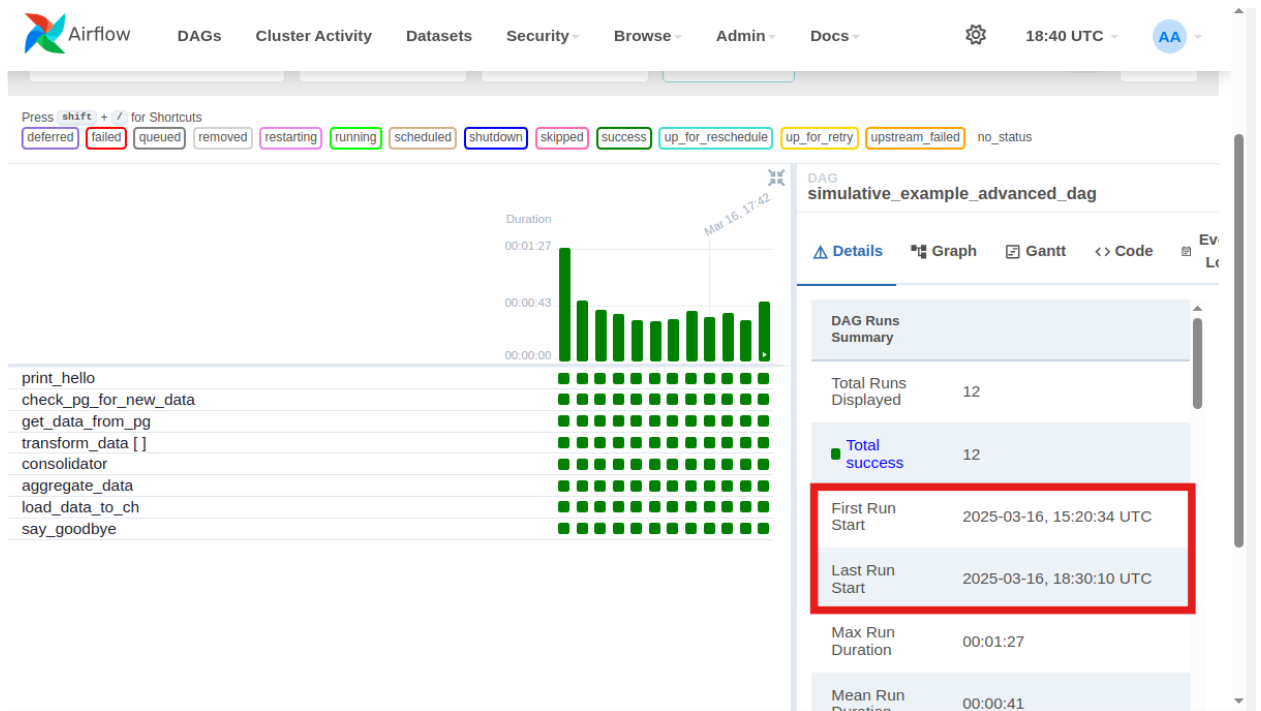
ПРЕДЫСЛОВИЕ

Для генерации данных из 1 задания с учетом технических ресурсов моего ноутбука, потребовалось 194 минуты (3 часа). Однако, на этом проблемы не закончились.

Позднее при выполнении 2 задания оказалось, что в моих условиях нереализуемо за один раз выполнить загрузку 97 строк данных из PostgreSQL в Clickhouse. Такая попытка, которая длилась больше часа, все равно закончилась крахом виртуальной машины:



Было принято решение вновь генерировать данные и загружать их в Clickhouse партиями, на это потребовалось еще около 3 часов (что можно видеть на скрине ниже). В связи с этим, 2 задание было сдано не вовремя, но хотя бы, оно сделано сквозь пот, боль и слезы.



ХОД РАБОТЫ

1. Самый оптимальный вариант генерации и загрузки данных из PostgreSQL в Clickhouse, при котором виртуальной машине хватает ресурсов для более-менее стабильной работы:

— установка автоматического выполнения дага с генерацией данных и загрузкой в PostgreSQL – каждые 1,7 минуты:

```

EXPLORER
...
simulative_example_advanced_dag.py
simulative_example_basic_dag.py x

LAB_0_WEBINAR
├── airflow
│   ├── config
│   └── dags
│       ├── __pycache__
│       ├── basic_dag.py
│       ├── simulative_example_advance...
│       ├── simulative_example_basic_da...
│       └── utils.py
├── data
│   ├── logs
│   ├── plugins
│   └── data/clickhouse/node1
│       ├── config.xml
│       ├── users.xml
│       └── fakerApi

airflow > dags > simulative_example_basic_dag.py
1 import datetime
2
3 from airflow.decorators import task, task_group
4 from airflow.models.dag import DAG
5
6
7 with DAG(
8     dag_id="simulative_example_basic_dag",
9     schedule=datetime.timedelta(minutes=1.7),
10    start_date=datetime.datetime(2025, 1, 1),
11    catchup=False,
12    tags=["simulative"],
13 ) as dag:
14
15     @task
16     def print_hello():
17         print("Hello, Simulative!")
18

```

— установка автоматического выполнения дага с загрузкой в Clickhouse – каждые 5 минут:

```

1 import datetime
2
3 from airflow.decorators import task
4 from airflow.models.dag import DAG
5
6
7 with DAG(
8     dag_id="simulative_example_advanced_dag",
9     schedule=datetime.timedelta(minutes=5),
10    start_date=datetime.datetime(2025, 1, 1),
11    catchup=False,
12    tags=["simulative"],
13 ) as dag:
14
15     @task
16     def print_hello():
17         print("Hello, Simulative!")

```

(До этого пробовались другие варианты выполнения, например 3 минуты и 18 минут соответственно, однако они ломались рано или поздно, либо в ClickHouse залетало только половина данных)

2. Далее вновь были запущены необходимые для работы Airflow контейнеры:

```

[+] Running 8/8
  ✓ faker-api Built 0.0s
  ✓ Network lab_0_webinar_default Created 0.5s
  ✓ Volume "lab_0_webinar_minio_data" Created 0.0s
  ✓ Container zookeeper Started 2.7s
  ✓ Container lab_0_webinar-minio-1 Started 3.7s
  ✓ Container pg Healthy 14.9s
  ✓ Container ch Healthy 35.0s
  ✓ Container lab_0_webinar-faker-api-1 Started 35.9s
● dev@dev-vm:~/Downloads/lab_etl/data_for_labs/lab_airflow/lab_0_webinar$ make up-af
sudo docker compose -f docker-compose-af.yaml up -d --build
[+] Running 48/48
  ✓ airflow-triggerer Pulled 117.3s
  ✓ airflow-init Pulled 117.3s
  ✓ airflow-scheduler Pulled 117.3s
  ✓ postgres Pulled 65.6s
  ✓ airflow-worker Pulled 117.3s
  ✓ redis Pulled 41.8s
  ✓ airflow-webserver Pulled 117.3s
[+] Running 1/1
  ✓ Volume "lab_0_webinar_postgres-db-volume" Created 0.0s
WARN[0117] Found orphan containers ([lab_0_webinar-faker-api-1 ch pg zookeeper lab_0_webinar-minio-1]) for thi
[+] Running 8/8u removed or renamed this service in your compose file, you can run this command with the --rem
  ✓ Volume "lab_0_webinar_postgres-db-volume" Created 0.0s
  ✓ Container lab_0_webinar-postgres-1 Healthy 20.6s
  ✓ Container lab_0_webinar-redis-1 Healthy 20.6s
  ✓ Container lab_0_webinar-airflow-init-1 Exited 56.8s
  ✓ Container lab_0_webinar-airflow-scheduler-1 Started 63.5s
  ✓ Container lab_0_webinar-airflow-worker-1 Started 59.3s
  ✓ Container lab_0_webinar-airflow-webserver-1 Started 64.1s
  ✓ Container lab_0_webinar-airflow-triggerer-1 Started 58.4s
○ dev@dev-vm:~/Downloads/lab_etl/data_for_labs/lab_airflow/lab_0_webinar$

```

Проверка доступа к Airflow (<http://localhost:8080>):

DAGs

UI for DAGs (Directed Acyclic Graphs) showing a list of DAGs and their status.

Buttons: All 3, Active 1, Paused 2, Running 0, Failed 0, ×simulative, Search DAGs

Auto-refresh: ☒ Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
<input checked="" type="checkbox"/> basic_dag simulative	airflow	0 18 ***	0 18 ***	2025-03-15, 18:00:00	2025-03-15, 18:00:00	
<input checked="" type="checkbox"/> simulative_example_advanced_dag simulative	airflow	41	0:05:00	2025-03-16, 23:35:00	2025-03-16, 23:40:00	
<input checked="" type="checkbox"/> simulative_example_basic_dag simulative	airflow	118	0:01:42	2025-03-16, 23:39:24	2025-03-16, 23:41:06	

3. Даги были поставлены на выполнение каждый со своей частотой срабатывания.

Успешно сгенерированные и загруженные данные в PostgreSQL (118 записей):

UI for PostgreSQL query execution and results display.

Query: `select * from public.person p;`

Results (118 rows):

id	Az name	123 age	Az address
108	Петухов Зосима Харлампьевич	96	п. Вендинга, бул. Пролетарский, д. 5/5 ст
109	Никитин Сила Игнатович	75	с. Шарья, пр. Ставропольский, д. 15 стр.
110	Тамара Афанасьевна Русакова	77	г. Пятигорск, пер. Виноградный, д. 475 с
111	Виктория Святославовна Гуляева	96	клх Дно, алл. Логовая, д. 3/7, 277246
112	Павел Фролович Марков	48	п. Льгов, наб. Кирпичная, д. 961 к. 492, 7
113	Агафья Александровна Овчинники	88	к. Качканар, пр. Медицинский, д. 851, 80
114	Блохина Полина Робертовна	28	ст. Калининград, ул. М.Горького, д. 4, 44
115	Виктория Кузьминична Колесники	57	к. Кош-Агач, ул. Терешковой, д. 3 стр. 89
116	Алевтина Эльдаровна Самойлова	89	ст. Светлогорск (Калин.), пр. Набережны
117	Варвара Владиславовна Сысоева	77	г. Сорочинск, ш. Радищева, д. 575, 31183
118	Ангелина Архиповна Брагина	23	ст. Саров (Морд.), ул. Бульварная, д. 1/3

118 row(s) fetched - 0.076s (0.013s fetch), on 2025-03-17 at 21:19:49

Успешно загруженные данные в Clickhouse:

<Airflow_0_pg> Script-2 <Airflow_0_ch> Script-3 ×

```

select 1 as value;

SELECT city, name
FROM `default`.person_count_by_city;

SELECT name
  
```

person_count_by_city 1 ×

SELECT city, name FROM `default` *Enter a SQL expression to filter results (use Ctrl+Space)*

	A-Z city	123 name
1	г. Горячинск	1
2	г. Казань	3
3	г. Красногорск (М	1
4	г. Курильск	1
5	г. Минеральные Е	2
6	г. Морозовск	1
7	г. Пятигорск	1
8	г. Северо-Куриль	1
9	г. Сорочинск	1
10	г. Тамбей	1
11	г. Усть-Кут	1
12	г. Хужир	1
13	д. Ачинск	1
14	д. Ачхой Мартан	1
15	д. Ачхой Мартан	1
16	д. Белорецк	1
17	д. Биробиджан	1

Refresh Save Cancel Export data 200 118

4. Далее необходимо найти топ-10 городов с наибольшим количеством людей. SQL-запрос:

```

SELECT city, name
FROM `default`.person_count_by_city
ORDER BY name DESC
LIMIT 10;
  
```

SQL Query:

```
SELECT city, name
FROM `default`.person_count_by_city
ORDER BY name DESC
LIMIT 10;
```

person_count_by_city 1 X

SQL Expression to filter results:

```
SELECT city, name FROM `default`
```

	A-Z city	123 name
1	к. Краснодар	4
2	с. Можайск	3
3	ст. Минеральные	3
4	г. Казань	3
5	ст. Новомичуринск	2
6	д. Ковров	2
7	клх Качканар	2
8	г. Минеральные Е	2
9	п. Тарко-Сале	1
10	п. Тольятти	1

5. Нахождение среднего количество людей в городах. SQL-запрос:

```
SELECT ROUND(AVG(name), 1) as avarege_amount_of_people
FROM `default`.person_count_by_city;
```

SQL Query:

```
SELECT ROUND(AVG(name),1) as avarege amount_of_people
FROM `default`.person_count_by_city;
```

person_count_by_city 1 X

SQL Expression to filter results (use Ctrl+Space):

```
SELECT ROUND(AVG(name),1) as
```

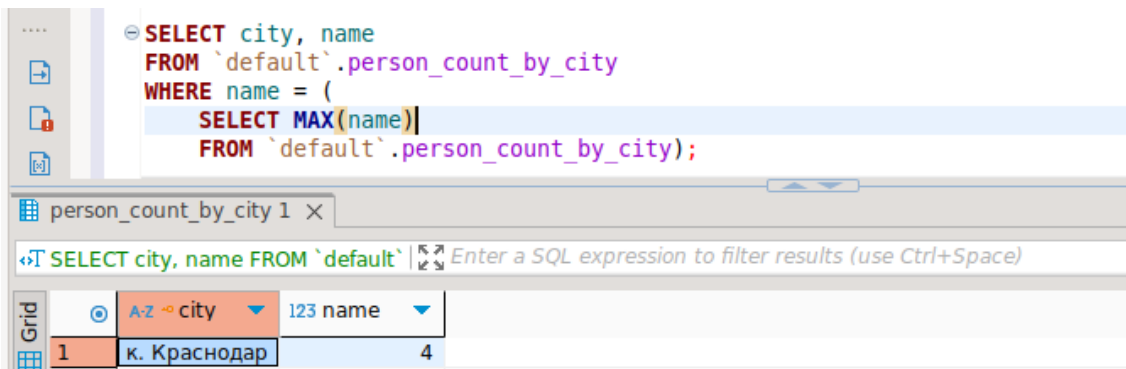
	123 avarege amount of people
1	1.1

6. Нахождение городов с максимальным и минимальным количеством людей. SQL-запрос:

```
SELECT city, name
FROM `default`.person_count_by_city
WHERE name = (
```



```
SELECT MAX(name)
FROM `default`.person_count_by_city);
```



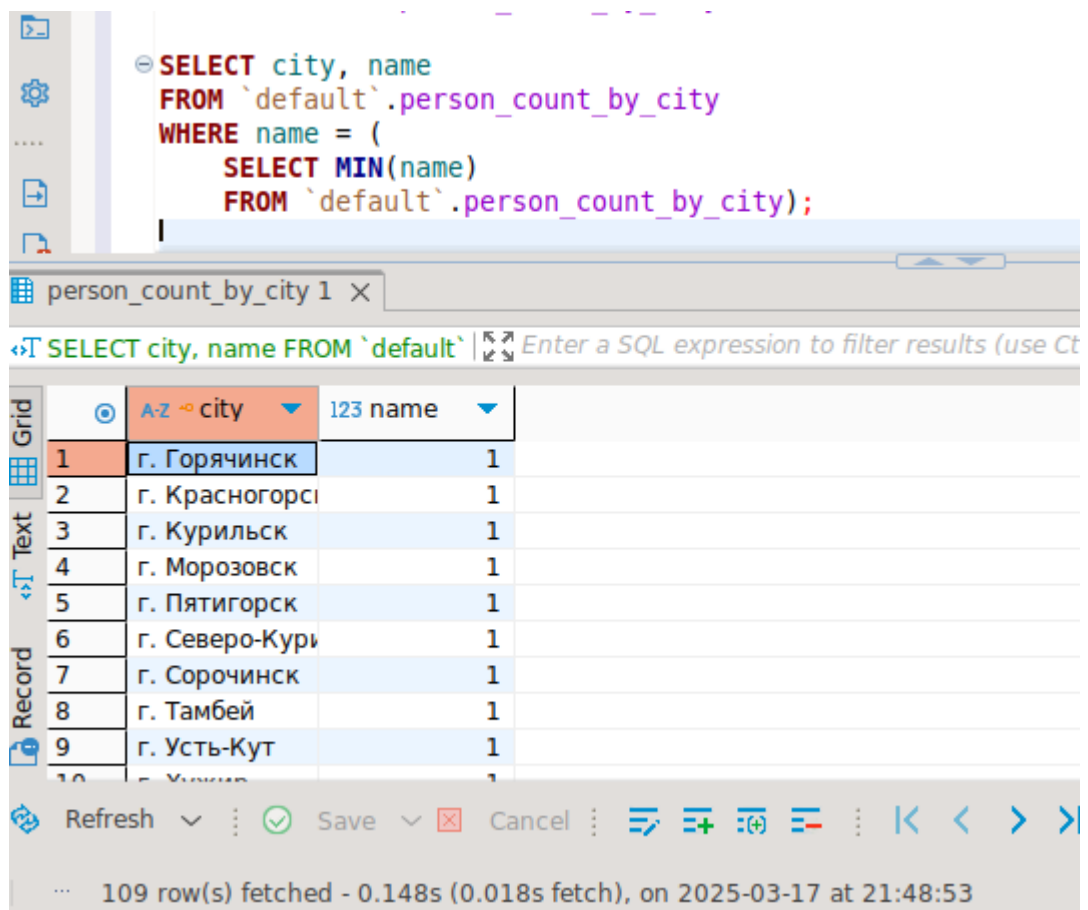
The screenshot shows a SQL IDE with a query editor and a results grid. The query is:

```
SELECT city, name
FROM `default`.person_count_by_city
WHERE name = (
    SELECT MAX(name)
    FROM `default`.person_count_by_city);
```

The results grid shows one row:

	A-Z city	123 name
1	к. Краснодар	4

```
SELECT city, name
FROM `default`.person_count_by_city
WHERE name = (
    SELECT MIN(name)
    FROM `default`.person_count_by_city);
```



The screenshot shows a SQL IDE with a query editor and a results grid. The query is:

```
SELECT city, name
FROM `default`.person_count_by_city
WHERE name = (
    SELECT MIN(name)
    FROM `default`.person_count_by_city);
```

The results grid shows 10 rows:

	A-Z city	123 name
1	г. Горячинск	1
2	г. Красногорск	1
3	г. Курильск	1
4	г. Морозовск	1
5	г. Пятигорск	1
6	г. Северо-Кури	1
7	г. Сорочинск	1
8	г. Тамбей	1
9	г. Усть-Кут	1
10	г. Усть-Кут	1

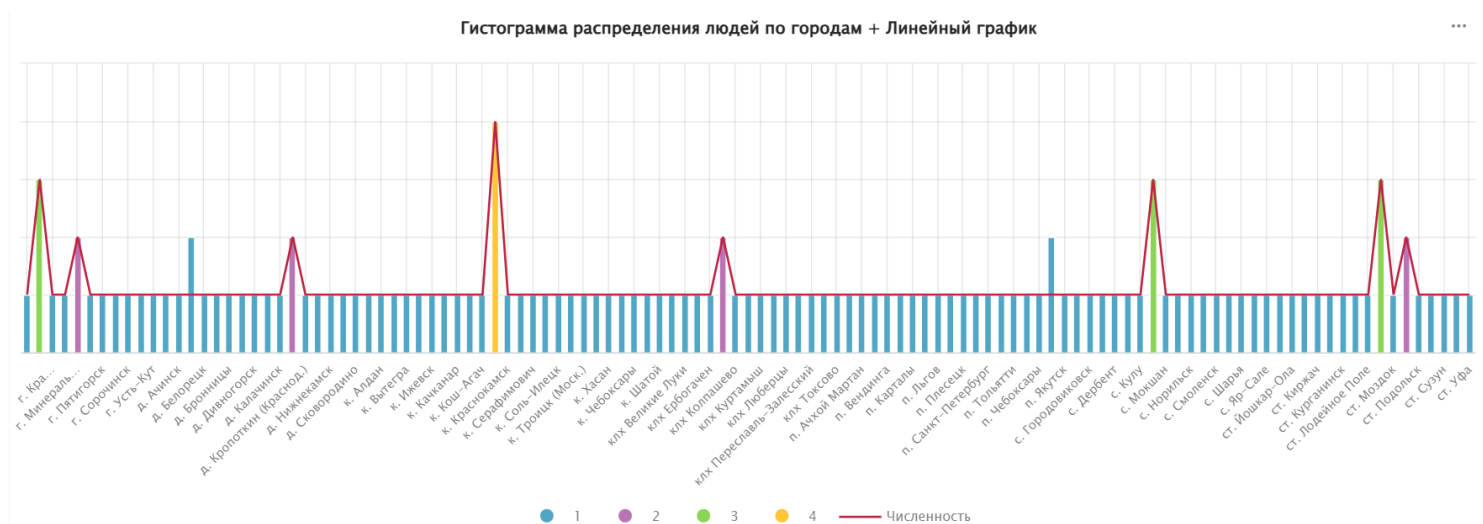
At the bottom, a status bar indicates: 109 row(s) fetched - 0.148s (0.018s fetch), on 2025-03-17 at 21:48:53

7. Для визуализации результатов, необходимо:

- Диаграмма топ-10 городов по количеству проживающих.



- Гистограмма распределения количества людей по городам.
- Линейный график, показывающий минимальное и максимальное количество людей в городах.



Для этого выгрузим файл с данными в формате CSV и визуализируем их с помощью YandexDataLens, потому что там быстро и удобно.

Ссылка на дашборд: <https://datalens.yandex/ii1x1mc381m04>

