

Санкт-Петербургский политехнический университет Петра Великого

Физико-механический институт



Институт компьютерных наук и кибербезопасности

Защита курсовой работы

по дисциплине

“Наука о данных и аналитика больших объемов информации”

на тему

“Анализ логистических данных транспортно-
экспедиторской компании для автоматизации расчёта
коммерческих предложений”



Зинкин Станислав

Неянин Вадим

Алмаметов Эмиль

Сафронов Герман

Санкт-Петербург, 2025

Актуальность проекта

- ❑ **Трудоёмкость и ошибки.** Подготовка коммерческого предложения требует обработки большого объема разнородных данных (груз, маршруты, тарифы, валюта), что при ручной работе ведёт к высоким затратам времени и риску ошибок;
- ❑ **Сложность оценки рентабельности.** Разрозненное хранение данных и отсутствие сводных расчётов затрудняют оперативную оценку себестоимости и рентабельности перевозок, а также сравнение альтернативных вариантов маршрутов;
- ❑ **Единая система анализа данных.** Использование аналитической платформы позволяет централизованно хранить данные по перевозкам и автоматизировать расчёт стоимости и ключевых показателей коммерческого предложения;
- ❑ **Скорость и прозрачность.** Автоматизация расчётов и единые правила обработки данных позволяют сократить трудоёмкость подготовки коммерческого предложения, повысить прозрачность расчётов и снизить влияние человеческого фактора;
- ❑ **Возможность роста.** Формирование аналитических отчётов и визуализаций по перевозкам создаёт основу для анализа динамики показателей, выявления узких мест и поддержки управленческих решений на уровне компании.



Цель и задачи

Цель: Сокращение трудозатрат на составление коммерческих предложений с помощью аналитических методов.

Основные задачи:

- ❑ Обработка и подготовка данных из 1С (Excel) и очистка дублирующихся и некорректных строк;
- ❑ Импорт подготовленных данных в Elasticsearch и создание индекса с корректными типами данных;
- ❑ Визуализация ключевых показателей деятельности: число грузов, вес, объём, выручка, маржа, расходы, загруженность менеджеров и логистов;
- ❑ Создание интерактивных дашбордов и аналитических панелей в Kibana с фильтрацией по датам, клиентам, странам, способам доставки и статьям затрат.



Цели аналитики

- ❏ Расчет минимальных, средних и медианных значений маржи, тарифа и иных параметров:
- ❏ Нахождение коэффициента сезонности по количеству накладных, объемам и весам перевозимых грузов:
- ❏ Расчет параметров с группировкой по направлениям, способам доставки, временным интервалам:
- ❏ Расчет коммерческого предложения.



Данные

Источник данных: Выгрузка из 1С (конфигурация «1АБ:Карго») в формате Excel.

Объем данных:

- 38 704 строк;
- 81 столбец;
- Период с 01.01.2025 по 30.06.2025;
- 11 Мб.

Особенности и проблемы данных:

- Дублирование строк по накладной и рейсу;
- Связь «одна накладная – несколько рейсов» и наоборот;
- Неструктурированные габариты груза (текстовый формат);
- Частично отсутствующие объемы и габариты;
- Разные валюты (RUB, USD, EUR, CNY), включая составные форматы;
- Зашифрованные справочники клиентов и сотрудников.

Накладная	Вес, кг	LDM	Число коробок	Габариты груза	Валюта страховой стоимости	Страховая стоимость
25-01-H00000000004	123,000	0,118		48*30*25cm8, 38*38*35cm, 32*22*12cm2, 44*30*35cm, 60*43*18cm, 52*62*41cm, 59*40*26cm, 48*30*14cm	USD	80 418,02
25-01-H00000000004	123,000	0,118		48*30*25cm8, 38*38*35cm, 32*22*12cm2, 44*30*35cm, 60*43*18cm, 52*62*41cm, 59*40*26cm, 48*30*14cm	USD	80 418,02
25-01-H00000000005	251,000	0,268				
25-01-H00000000005	251,000	0,268				
25-01-H00000000005	251,000	0,268				
25-01-H00000000005	251,000	0,268				
25-01-H00000000005	251,000	0,268				
25-01-H00000000005	251,000	0,268				
25-01-H00000000006	566,000	0,400	120x80x74		руб.	393 871,93
25-01-H00000000006	566,000	0,400	120x80x74		руб.	393 871,93
25-01-H00000000006	566,000	0,400	120x80x74		руб.	393 871,93
25-01-H00000000006	566,000	0,400	120x80x74		руб.	393 871,93
25-01-H00000000006	566,000	0,400	120x80x74		руб.	393 871,93

Рисунок 1. Пример данных



Архитектура решения

Используемый стек:

- ❑ ELK Stack: Elasticsearch + Kibana;
- ❑ Python для обработки и загрузки данных;
- ❑ Docker Compose для контейнеризации и развертывания компонентов.

Компоненты системы:

- ❑ Контейнер Elasticsearch — хранение и быстрый поиск логистических данных;
- ❑ Контейнер Kibana — визуализация показателей и аналитические дашборды;
- ❑ Контейнер Python-приложения — подготовка данных, расчёт метрик и загрузка в Elasticsearch.

Поток данных:

Excel (выгрузка из 1C) → CSV (очистка и нормализация данных) → NDJSON (формат для bulk-загрузки) → Elasticsearch (индексация данных) → Kibana (дашборды и аналитика).



Подготовка данных

Очистка и нормализация данных:

- ❑ Удаление дубликатов строк по связке накладная – рейс;
- ❑ Приведение числовых полей к единому типу (вес, объём, LDM);
- ❑ Обработка пропусков и некорректных значений;
- ❑ Нормализация текстовых полей (города, страны, форматы дат).

Пример преобразования данных:

Показатель	До обработки	После обработки
Габариты груза	46x41x31 cm (6)	0.116 LDM
Объём	Текстовое поле	0.578 CBM
Валюта	USD / EUR / CNY	RUB



Оценка данных

- **Неполнота**

В некоторых используемых столбиках отсутствует ~20% данных.

- **Качество**

Столбик 'Габариты груза', используемый для расчета коммерческого предложения содержит ~1000 вариаций (~3% от общего кол-ва строк) записи габаритов. Названия городов написаны по-разному.

- **Несогласованность**

Ячейки столбика 'Габариты груза' иногда содержат данные другого столбца, что приводит в итоге в несовпадению.

☑ "109*67*220 / 165,5 кг210*98*141 / 198 кг155*68*210 / 335,5 кг155*68*210 / 244,5 кг155*68*210 / 249 кг155*68
☑ "116*81*79 / 1236,5 кг117*81*80 / 1234,5 кг117*82*79 / 1236 кг116*81*80 / 1235,5 кг118*87*80 / 1237 кг116*81
☑ "122*82*84122*82*8498*84*87122*84*64109*96*82106*77*103106*91*100122*82*85122*82*8512
☑ "142*145*143 cm142*145*143 cm136*139*133 cm130*65*83 cm202*81*153 cm
☑ "1pal. -1,3*0,84*1,838pal. -1,2*0,8*1,15"
☑ "2 ящика - 440*280*3406 ящиков - 440*310*340"
☑ "2*0.91*1.51.2*0.8*1.91.3*0.84*1.451.2*0.8*1.221.2*0.81*1.81.3*0.83*2.11.2*0.8*1.221.31*0.83*1.441.31*0.84*2.1"
☑ "435*170*188/ 2100 кг226*116*192/ 360 кг287*136*146/ 600 кг216*136*143/ 420 кг225*200*150/ 480 кг"
☑ "61*52*72 / 63,5 кг200*215*144 / 404,5 кг205*200*145 / 400 кг "
☑ "6pal. - 1.21*0.84*1.43 1pal. -1.21*0.84*0.99 "
☑ "6pal. -1.21*0.84*1.431pal. -1.21*0.84*0.99 1pal. -1.21*0.84*0.76"
☑ (120x80x77)см 211кг (120x80x77)см 216кг (120x80x58)см 159кг
☑ (16)82x64x64
☑ (2)120*80*167см, 120*80*162см, 120*80*85см
☑ (2)120x80x114
☑ (2)120x80x55 120x80x55
☑ (2)123x83x120/123x83x90/133x128x88

Рисунок 2. Вариации габаритов



Применяемые методы при обработке

❑ Регулярные выражения (всего 12)

Некоторые группы габаритов отдаленно имеют одинаковый формат;

Для размеров вида 100x50x30 см (5):

```
r"(\d+[,.]?*\d*)\s*[*x]\s*(\d+[,.]?*\d*)\s*[*x]\s*(\d+[,.]?*\d*)\s*(?:cm|m)?\s*\s*(\d+[,.]?*\d*)"
```

Для размеров вида 5 паллет 120x80x90:

```
r"(\d+)\s*(?:палл|кор|шт|pallet|box)\.?(?:\d+)[*x]\s*(\d+[*x]\s*(\d+))"
```

❑ XML-запросы в ЦБ РФ

Для конвертации валюты используется дата накладной и данные из ЦБ РФ;

❑ Словари замены

Транслитерация некоторых названий городов происходит неверно, есть необходимость использования словарей.

```
▼<ValCurs ID="R01235" DateRange1="01.01.2025"
  ▼<Record Date="10.01.2025" Id="R01235">
    <Nominal>1</Nominal>
    <Value>102,2911</Value>
    <VunitRate>102,2911</VunitRate>
  </Record>
  ▼<Record Date="11.01.2025" Id="R01235">
    <Nominal>1</Nominal>
    <Value>101,9146</Value>
    <VunitRate>101,9146</VunitRate>
  </Record>
  ▼<Record Date="14.01.2025" Id="R01235">
    <Nominal>1</Nominal>
    <Value>102,7081</Value>
    <VunitRate>102,7081</VunitRate>
  </Record>
```

Рисунок 3. XML-запрос в ЦБ РФ

Импорт данных в Elasticsearch

```
# If we do not specify the date type for individual fields,  
# they will be of the text/keywords type.  
mapping = {  
  "mappings": {  
    "dynamic": True, # the remaining fields are determined automatically  
    "properties": {  
      "Дата накладной": {"type": "date", "format": "dd.MM.yyyy"},  
      "Плановая дата начала": {"type": "date", "format": "dd.MM.yyyy"},  
      "Плановое время начала": {"type": "date", "format": "HH:mm"},  
      "Плановая дата завершения": {"type": "date", "format": "dd.MM.yyyy"},  
      "Плановое время завершения": {"type": "date", "format": "HH:mm"},  
      "Фактическая дата начала": {"type": "date", "format": "dd.MM.yyyy"},  
      "Фактическое время начала": {"type": "date", "format": "HH:mm"},  
      "Фактическая дата завершения": {"type": "date", "format": "dd.MM.yyyy"},  
      "Фактическое время завершения": {"type": "date", "format": "HH:mm"},  
    },  
  },  
}
```

Рисунок 4. Задание mapping для полей даты и времени

Без явного mapping поля с датами и временем автоматически интерпретируются как текстовые, что делает невозможным использование временных фильтров и временных агрегаций.

Импорт данных в Elasticsearch

Search: Время × Field type: 8 Schema type: 2 Refresh Add field

Name ↑	Type ↓	Format ↓	Search... ↓	Aggreg... ↓	Exclud... ↓	Actions
Плановое время завершения	text		•			✎
Плановое время завершения.keyword	keyword		•	•		✎
Плановое время начала	text		•			✎
Плановое время начала.keyword	keyword		•	•		✎
Фактическое время завершения	text		•			✎
Фактическое время завершения.keyword	keyword		•	•		✎

Рисунок 5. Без mapping

Fields (4 / 149) Field filters (0) Relationships (47)

Search: Время × Field type: 9 Schema type: 2 Refresh Add field

Name ↑	Type ↓	Format ↓	Search... ↓	Aggreg... ↓	Exclud... ↓	Actions
Плановое время завершения	date		•	•		✎
Плановое время начала	date		•	•		✎
Фактическое время завершения	date		•	•		✎
Фактическое время начала	date		•	•		✎

Рисунок 6. С mapping

Импорт данных в Elasticsearch

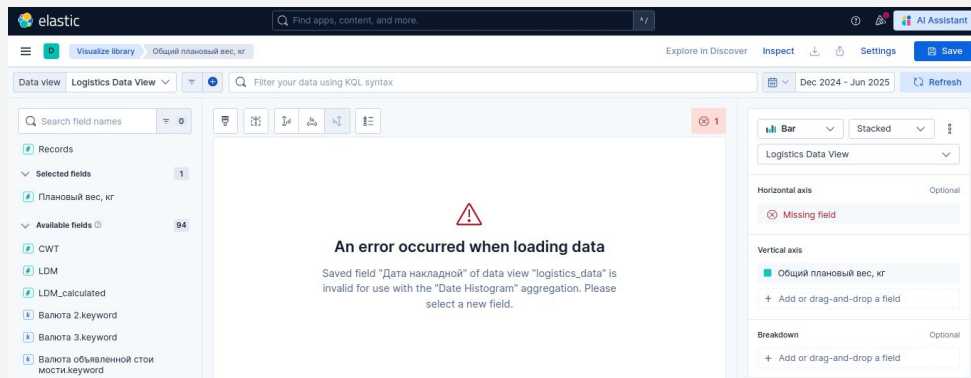


Рисунок 7. Без mapping



Рисунок 8. С mapping

Импорт данных в Elasticsearch

```

# Indexing documents using Bulk API
actions = []
success_count = 0
error_count = 0

for i, line in enumerate(lines, start=1):
    try:
        doc = json.loads(line)
        actions.append({"_index": INDEX_NAME, "_source": doc})
    except json.JSONDecodeError:
        print(f"\t[{i}] Invalid JSON, skipped.")
        error_count += 1

# Execute bulk indexing
success_count, errors = helpers.bulk(
    es, actions, chunk_size=CHUNK_SIZE, raise_on_error=False
)
error_count = len(actions) - success_count
```

Рисунок 9. Загрузка данных в Elasticsearch с использованием Bulk API

Поштучная индексация создаёт большое количество HTTP-запросов, что резко снижает производительность при больших объёмах данных. Bulk API решает эту проблему за счёт пакетной обработки.

Визуализация в Kibana

```

# Open the NDJSON file in binary mode
# Important: requests.post with files=... automatically forms multipart/form-data
with open(FILE_PATH, "rb") as f:
    files = {"file": (FILE, f, "application/ndjson")}

# POST request to the Saved Objects import API
# Parameter overwrite=true allows overwriting objects if they already exist
r = requests.post(
    f"{KIBANA_URL}/api/saved_objects/_import?overwrite=true",
    headers=headers,
    files=files, # pass the file via multipart/form-data
)

# Print the result of the request
print("\tStatus code:", r.status_code, end="\n\n")
```

Рисунок 10. Импорт сохранённых объектов в Kibana с использованием Saved Objects API

Сохранённые объекты Kibana представляют собой набор настроек визуализаций и дашбордов. Их импорт позволяет быстро развернуть готовую систему визуализации на новых данных или в новом окружении.

Визуализация в Kibana

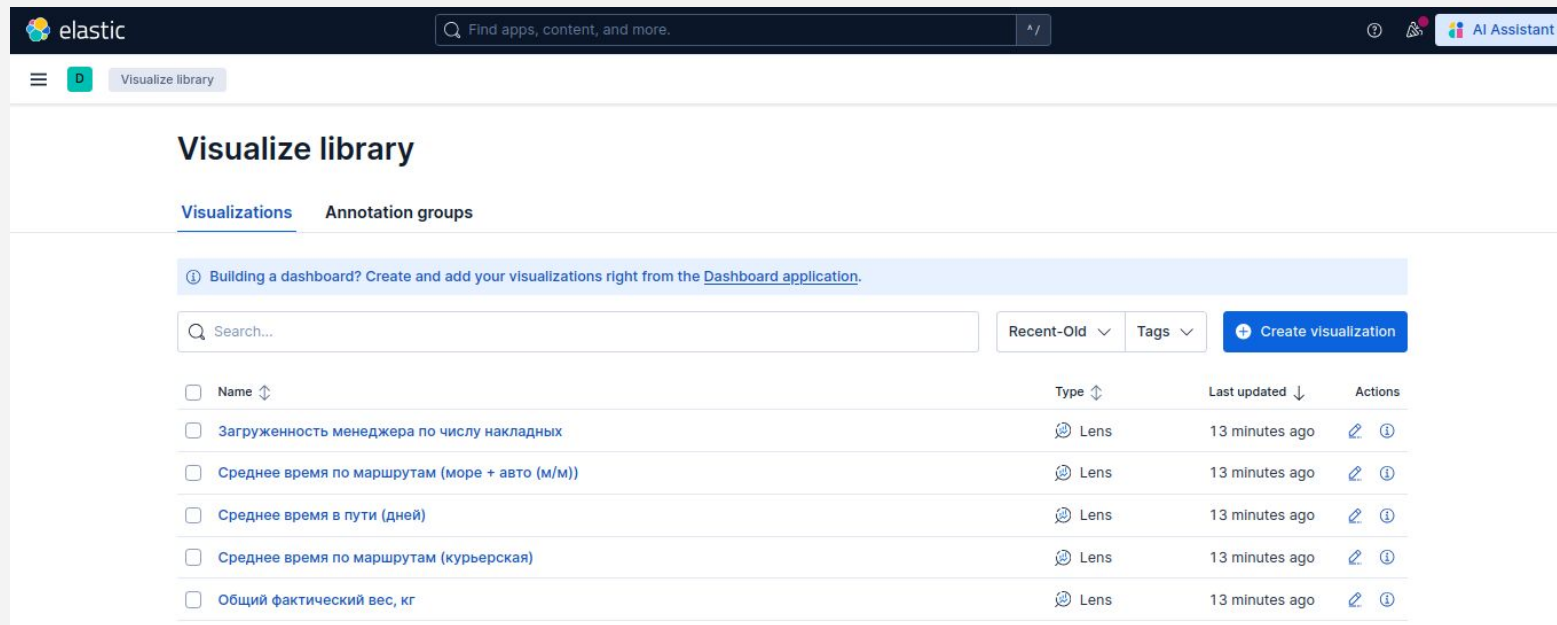


Рисунок 11. Сохранённые объекты: графики

Визуализация в Kibana

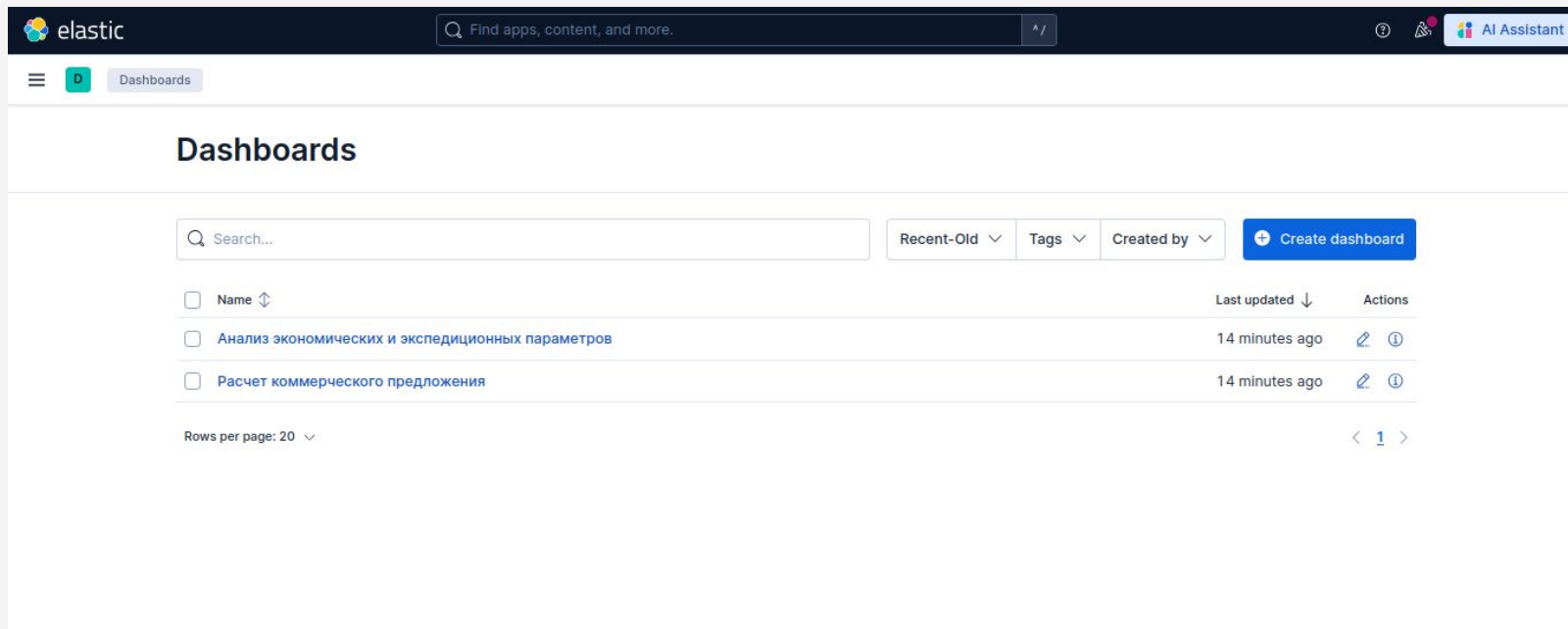


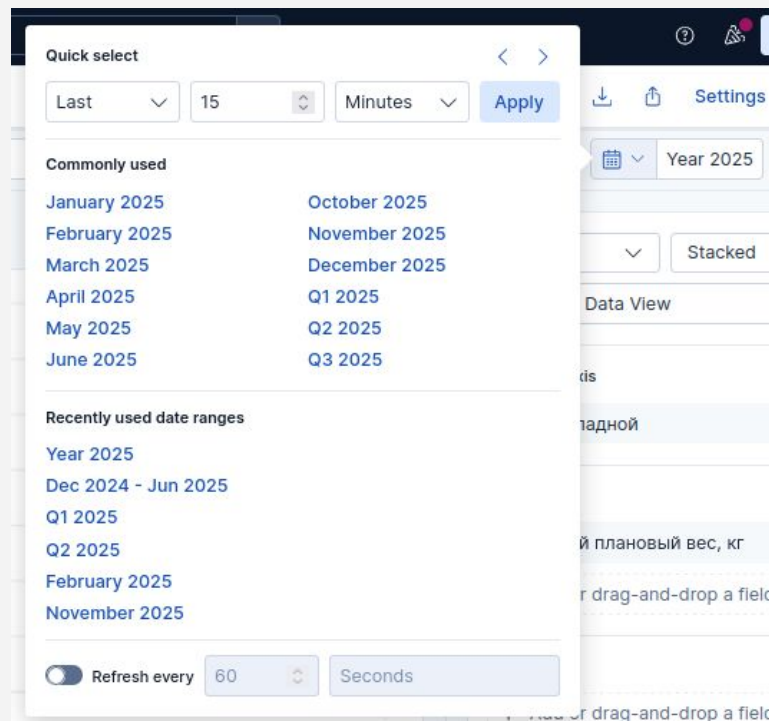
Рисунок 12. Сохранённые объекты: дашборды

Визуализация в Kibana



Рисунок 13. Сохранённые объекты: временной фильтр по умолчанию

Визуализация в Kibana



**Рисунок 14. Сохранённые объекты:
временные фильтры быстрого доступа**

Визуализация в Kibana

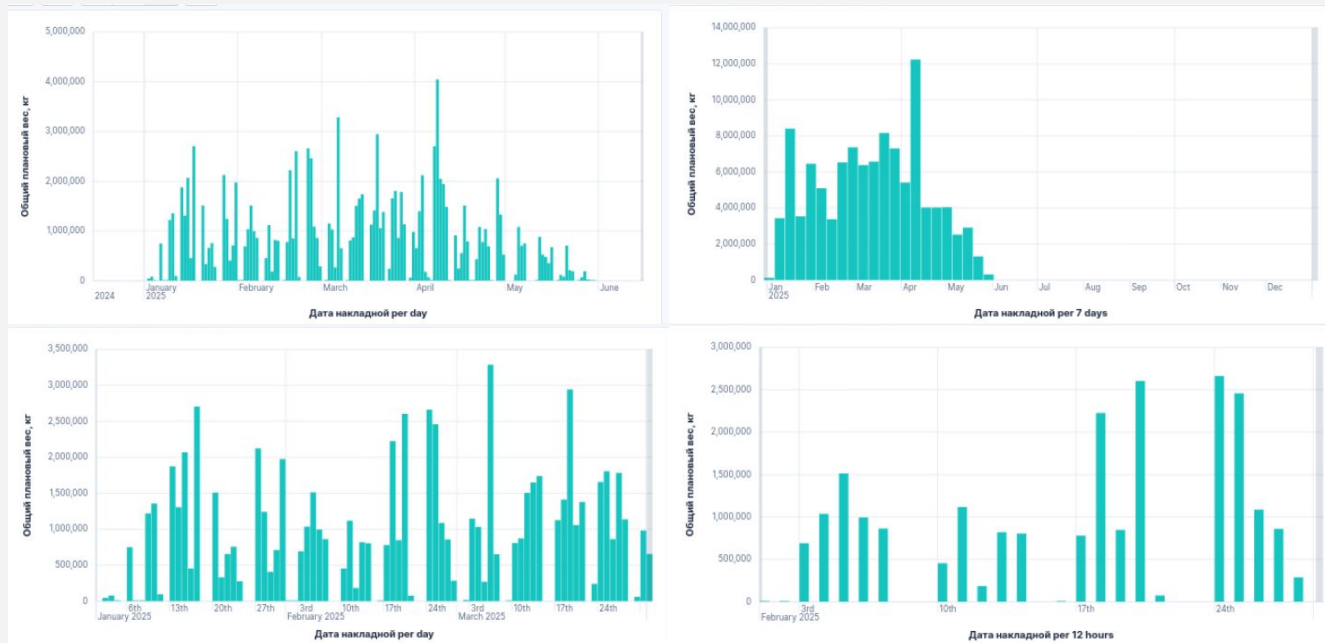


Рисунок 15. Один и тот же график, но с разными временными интервалами

Визуализация в Kibana. Vega

- Создано **57** уникальных визуализаций.
- Визуализации на картинке ниже созданы с помощью **Vega**.

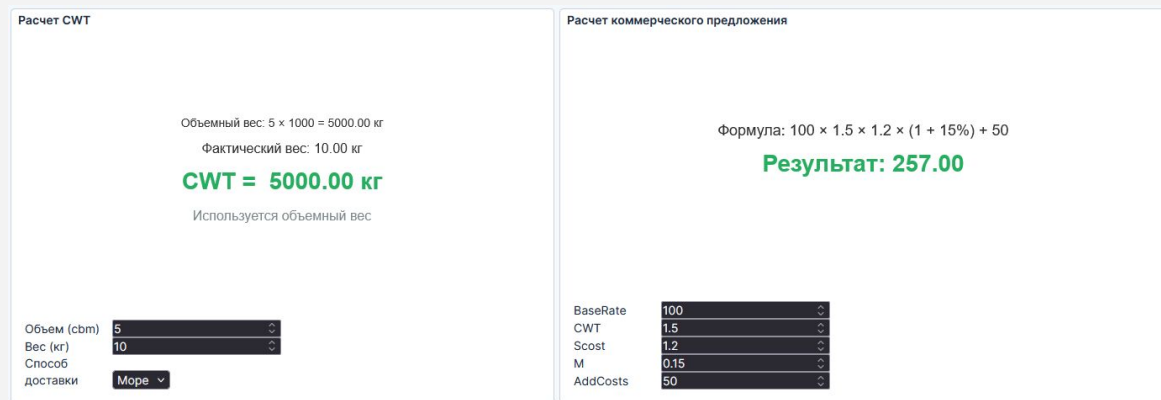


Рисунок 16. Интерактивные визуализации расчета объемного веса и коммерческого предложения

```
37  "marks": [  
38  {  
39    "type": "text",  
40    "encode": {  
41      "enter": {  
42        "align": {"value": "center"},  
43        "baseline": {"value": "middle"},  
44        "fontSize": {"value": 18},  
45        "fill": {"value": "#333"}  
46      },  
47      "update": {  
48        "text": {  
49          "signal": "'Формула: ' + baseRate + ' × ' + cwt + '  
                    × ' + scost + ' × (1 + ' + (m * 100) + '%)' + ' +  
                    addCosts'"  
50        },  
51        "x": {"signal": "width / 2"},  
52        "y": {"signal": "height / 2 - 40"}  
53      }  
54    }  
55  ],
```

Рисунок 17. Фрагмент Vega-кода

Визуализации в Kibana. Lens

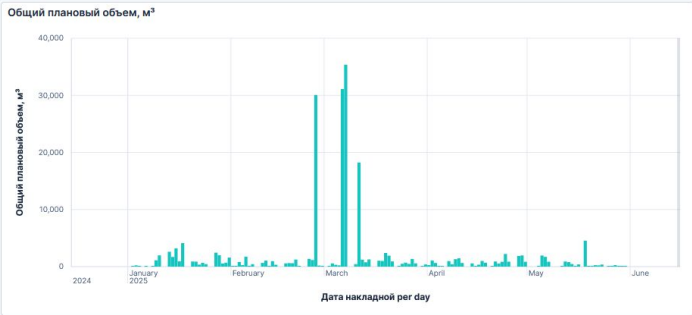


Рисунок 18. Зависимость общего планового объема от даты накладной

Коэффициенты сезонности

Месяц per month	Коэфф. сезонности по объему	Коэфф. сезонности по кол-ву дней доставки	Коэфф. сезонности по весу	Коэфф. сезонности по кол-ву накладных
2024-12-01	(null)	(null)	(null)	0
2025-01-01	0.655	1.047	0.892	1.23
2025-02-01	0.878	1.22	1.097	1.205
2025-03-01	1.882	0.991	1.232	1.469
2025-04-01	0.993	1.051	1.379	1.654
2025-05-01	0.592	0.691	0.4	1.442
2025-06-01	(null)	(null)	(null)	0

Рисунок 20. Коэффициенты сезонности по разным показателям



Рисунок 19. Зависимость доли каждого вида транспорта от даты накладной



Рисунок 21. Загруженность менеджеров по числу накладных

Docker Compose и автоматизация

Docker Compose используется для оркестрации всех компонентов аналитического решения и запуска окружения одной командой.

Автоматизация запуска:

- ❑ Все сервисы поднимаются одной командой `docker compose up -d`;
- ❑ Python-приложение ожидает готовности Elasticsearch и Kibana;
- ❑ Полный пайплайн (подготовка данных → загрузка → визуализация) выполняется без участия пользователя.

Рисунок 22. Полный пайплайн

```
=====
[Starting python app]
=====
Container python-app Creating
Container python-app Created
Attaching to elasticsearch, kibana, python-app
python-app | [Waiting for Elasticsearch to be fully ready...]
python-app | [Starting Python application...]
python-app | [Excel Export Processing]
python-app | Export dates: 01.01.2025 - 31.05.2025
python-app |
python-app | [Currency Conversion]
python-app | Fetching currency rates...
python-app | Currency rates successfully fetched
python-app | Conversion completed
python-app |
python-app | [LDM Calculation]
python-app | 25% of rows processed
python-app | 50% of rows processed
python-app | 75% of rows processed
python-app |
python-app | [Saving Results]
python-app | Saving processed data -> data/prepared_data.csv
python-app | Saving unprocessed dimensions -> data/bad_gabarit.csv
python-app | Saving null LDM -> data/null_ldm.csv
python-app | Saving statistics -> data/stat.log
python-app |
python-app | [CSV -> NDJSON]
python-app | Reading CSV: data/prepared_data.csv
python-app | Successfully loaded CSV. Total rows: 33059
python-app | Done! NDJSON saved as: data/prepared_data.json
python-app |
python-app | [Elasticsearch]
python-app | Connected to Elasticsearch.
python-app |
python-app | Index 'logistics_data' already exists. Deleting it...
python-app | Index 'logistics_data' deleted.
python-app |
python-app | Creating an index 'logistics_data'...
python-app | Index 'logistics_data' created.
python-app |
python-app | Opening data file: data/prepared_data.json
python-app | Found 33059 lines to load.
python-app |
python-app | Data loading completed.
python-app | Successfully loaded documents: 33059
python-app | Errors: 0
python-app | Index name: logistics_data
python-app |
python-app | [Kibana]
python-app | Kibana is fully ready!
python-app | Importing saved objects...
python-app | Status code: 200
python-app |
python-app | [Total execution time: 159.82 s]
python-app exited with code 0
```

Docker Compose и автоматизация

The screenshot shows the GitLab web interface for a repository named 'data-analysis'. At the top, there are tabs for 'main' and 'data-analysis', with 'data-analysis' being the active branch. Below the repository name, there's a commit message 'Новый ndjson и батник' by 'spbtu-admin' from 3 days ago, along with a commit hash '268bd3b9' and a 'History' link. The main part of the page is a table listing the repository's files and folders, including their last commit messages and update times. At the bottom, there's a section for 'README.md' titled 'Logistics Data Pipeline' with a brief description of its functionality.

Name	Last commit	Last update
config	Reduce chunk_size in config.yml to 10000 to prevent ES ...	4 weeks ago
data	Новый ndjson и батник	3 days ago
models	BaseParams naming fix	1 month ago
scripts	Добавлен расчет комм предложения, время подсчета ...	1 week ago
utils	Use .iloc to remove pandas FutureWarning	4 weeks ago
.gitignore	Add scripts/csv_to_ndjson.py	1 month ago
LICENSE	Add MIT LICENSE	1 month ago
README.md	Update README.md	1 month ago
docker-compose.yml	реестр	2 weeks ago
install_containers.bat	Добавлен расчет комм предложения, время подсчета ...	1 week ago
requirements.txt	Add PyYAML to requirements.txt	4 weeks ago
start.bat	Новый ndjson и батник	3 days ago
start.sh	Edit start.sh	2 weeks ago

Logistics Data Pipeline
Processes logistics data from Excel and loads it into Elasticsearch and Kibana.

Рисунок 23. Репозиторий GitLab

Docker Compose и автоматизация

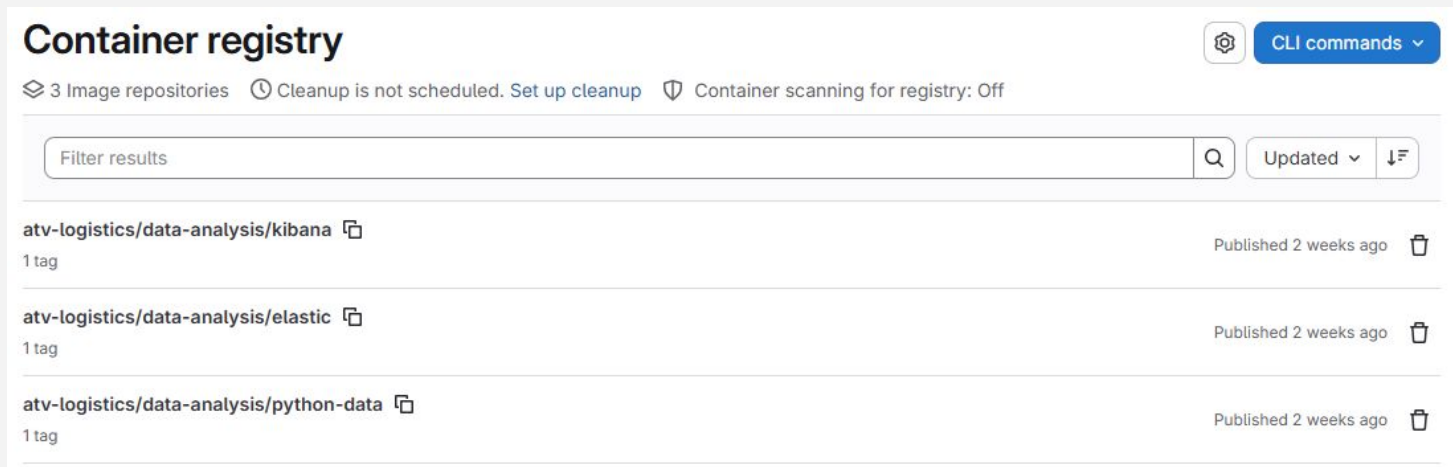


Рисунок 24. Образы Elasticsearch, Kibana и Python в Gitlab Container Registry

Горизонтальное масштабирование

Elasticsearch обладает возможностью линейного масштабирования.



Чтение

Механизм: Запросы выполняются параллельно на всех шардах

Результат: + узлы/реплики → + QPS (~линейно);



Запись

Механизм: Данные пишутся в разные первичные шарды

Результат: + первичные шарды → + скорость индексирования;



Хранение

Механизм: Шарды распределяются по новым узлам

Результат: + узлы с дисками → + общий объём кластера.



Конечный вид дистрибутива

- ❑ Объем ~ 1.3 Гб;
- ❑ Архивы контейнеров;
- ❑ Скрипт запуска start.bat;
- ❑ Папка data с исходной выгрузкой и результатами.

```
C:\Windows\system32\cmd.exe

[Checking Docker Images]
=====
Checking for python-data:3.13.9-ver-1...
Image python-data:3.13.9-ver-1 found
Checking for elasticsearch:9.2.0...
Image elasticsearch:9.2.0 found
Checking for kibana:9.2.0...
Image kibana:9.2.0 found

All required images are already available.

[Starting Services]
=====
[+] Running 3/3
✓ Container kibana          Removed
                           0.0s
✓ Container elasticsearch   Removed
                           0.1s
✓ Network data_for_zip_elastic Removed
                           0.3s

python-app
[+] Running 3/3
✓ Network data_for_zip_elastic Created
                           0.1s
✓ Container elasticsearch   Started
                           1.0s
✓ Container kibana          Started
                           1.2s
```

```
C:\Windows\system32\cmd.exe

[Elasticsearch starting on http://localhost:9200]
[Kibana starting on http://localhost:5601]

[Waiting 10 seconds]
[Starting python app]
=====
Container python-app Creating
Container python-app Created
Attaching to elasticsearch, kibana, python-app
python-app | [Waiting for Elasticsearch to be fully ready...]
python-app | [Starting Python application...]
python-app | [Excel Export Processing]
python-app |       Export dates: 01.01.2025 - 31.05.2025
python-app |
python-app | [Currency Conversion]
python-app |       Fetching currency rates...
python-app |       Currency rates successfully fetched
python-app |       Conversion completed
python-app |
python-app | [LDM Calculation]
python-app |       25% of rows processed
python-app |       50% of rows processed
python-app |       75% of rows processed
python-app |
python-app | [Saving Results]
python-app |       Saving processed data -> data/prepared_data.csv
python-app |       Saving unprocessed dimensions -> data/bad_gabarit.csv
python-app |       Saving null LDM -> data/null_ldm.csv
```

```
.
├── containers
│   ├── elastic.tar
│   ├── kibana.tar
│   └── python.tar
├── data-analysis
│   ├── config
│   │   └── config.yml
│   ├── data
│   │   ├── export.ndjson
│   │   └── report.xlsx
│   ├── docker-compose.yml
│   ├── install_containers.bat
│   ├── LICENSE
│   ├── models
│   │   ├── BaseParams.py
│   │   └── Invoice.py
│   ├── README.md
│   ├── requirements.txt
│   └── scripts
│       ├── calculate_params.py
│       ├── config_loader.py
│       ├── csv_to_ndjson.py
│       ├── generate_test_data.py
│       ├── import_kibana_objects.py
│       ├── load_data_to_es.py
│       ├── main.py
│       └── prepare_data.py
├── start.bat
├── start.sh
├── utils
│   ├── converters
│   │   ├── model_converters.py
│   │   └── type_converters.py
│   └── debug_tools.py
└── 9 directories, 26 files
```

Результаты проекта

- ❑ Подготовлен дистрибутив и написана документация;
- ❑ Написано **~1000** строк программного кода, сделано **67** коммитов;
- ❑ Проведено **6** встреч по **~40** минут;
- ❑ Построено **57** уникальных визуализаций;
- ❑ Рассчитаны основные экономические показатели;
- ❑ Приложение работает в контейнерах;
- ❑ Реализован автоматический запуск по скрипту;
- ❑ Обработка выгрузки за полгода занимает **~150** с.;
- ❑ Внедрена возможность выполнения расчета коммерческого предложения.



Спасибо за внимание!
Готовы ответить на вопросы.

