ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Высшая школа Бизнеса

*Ефимов Тимофей Денисович, Мелехин Матвей Дмитриевич, Могушкова Фатима Магомедовна, Хисамутдинова Эмилия Рустамовна, Яшина Ева Георгиевна*

# “Проектирование базы данных компании - “*TechnoLogist*”, занимающейся грузоперевозками и логистикой”

КОНТРОЛЬНАЯ ДОМАШНЯЯ РАБОТА

по БД

образовательная программа «Бизнес-информатика»

Москва 2025

## Данные расположения базы данных:

## params = urllib.parse.quote\_plus(

## "DRIVER={ODBC Driver 17 for SQL Server};"

## "SERVER=51.250.22.137,1433;"

## "DATABASE=magistral\_2025\_kdz;"

## "UID=student;"

## "PWD=sql2023BI;"

## "TrustServerCertificate=yes;")

Распределение задач:

1)Выбор предметной темы - Хисамудинова Эмилия, Мелехин Матвей

2)Обоснование нужды в БД - Хисамудинова Эмилия, Ефимов Тимофей

3)Возможные сценарии с клиентами - Могушкова Фатима, Яшина Ева

4)Список сущностей + связи + атрибуты - Могушкова Фатима, Яшина Ева

5)Бизнес-функции - Яшина Ева, Могушкова Фатима, Хисамудинова Эмилия

6) Инфологическая, даталогическая модели, 3 NF - Яшина Ева, Могушкова Фатима, Хисамудинова Эмилия

7) создания БД - Ефимов Тимофей, Мелехин Матвей

7)Запросы к базе данных - Мелехин Матвей, Ефимов Тимофей

8) Дашборды - Ефимов Тимофей, Мелехин Матвей

## Выбор предметной темы.

Тема, выбранная нашей командой для работы - создание сервиса, автоматизирующего грузоперевозки и транспортную логистику: перенос логистики грузоперевозок в облако (и базу данных в ней) для снижения затрат предприятий.

Конечный потребитель: владельцы и участники бизнеса, нуждающегося в функционале приложения.

Отрасль: IT

Географический рынок: Международный(СНГ)

Главными направлениями деятельности компания определяет(Продукты/услуги.):

1. управление движения заказов на грузоперевозку
2. автоматизация документооборота
3. автоматизация аналитики
4. поддержание мобильного приложения
5. сопровождение потребител

## Обоснование нужды в БД

Перенос услуг логистики в онлайн помогает бизнесу избежать таких проблем, как

* Человеческий фактор и производные от него ошибки из-за наличия ручного труда в области организации логистики. Например, один из участников договора оговорился в телефонном разговоре, из-за чего возник конфликт о времени загрузки.
* Отсутствие точного контроля за грузоперевозками со стороны отправителей, срыв сроков. Например, при отправлении груза и попадания водителя в длительную пробку, принимающая сторона может не быть своевременно уведомлена о переносе времени отгрузки.
* Процесс грузоперевозки тяжело анализировать экспедиторам из-за того, что при классической схеме: договор о времени загрузки и отгрузке, а также цены; тяжело собирать аналитику. Фактически, это приходится замерять вручную, принимая возможность погрешностей и неточностей. Например, средняя цену перевозки груза на километр пути придется считать как цена заказа делить на самостоятельно подсчитанное расстояние пути, тогда как длинну этого пути можно узнать лишь примерно, из-за незнания фактического маршрута перевозчика.

Учет всего процесса перевозки в базе данных, которая собирает и агрегирует спрос и предложение на грузоперевозки поможет избавиться от всех вышеперечисленных трудностей.

Несмотря на то, что это современный подход к транспортировке, которым пользуется не каждая компания, этот сервис нельзя назвать инновацией, так как уже существуют такие сервисы как fura, GrooseGo, Яндекс.Магистрали.

## Возможные сценарии с клиентами

Клиент: ООО “ДомоСтрой”(строительная компания) - заинтересована в транспортировке стройматериалов по территории РФ

До использования предлагаемого сервиса - проблема с логистикой в связи с ручным управлением перевозками, долгий поиск нужного водителя, трудная аналитика в связи с разрозненными данными, срывы сроков из-за отсутствия контроля

Пример создания заказа:

1. Выбирает параметры:
2. Груз: 10 тонн кирпича.
3. Маршрут: Москва - Кунгур
4. Срок: 48 часов.

Предложение доставки от системы:

- Система предлагает 5 вариантов перевозчиков с рейтингом

Заключение договора

- После выбора перевозчика система генерирует договор, стороны подписывают его электронно(можно коллабу с КЭДО)

Мониторинг и отслеживание

- Груз отслеживается на карте. При задержке из-за пробки/поломки/нештатного поведения система уведомляет «ДомоСтрой» и автоматически корректирует время разгрузки.

Заключение

- После доставки формируется акт, перевозчик получает оплату, клиент оставляет отзыв.

## Список сущностей + связи + атрибуты

При выявлении сущностей были выделены:

* **Customer**

*Описание:*

Непосредственно отправитель груза. Он вносит оплату, отслеживает местоположение товара. Может быть фабрикой, складским центром, просто ИП.

*Поля:*

* id: PK
* name: название компании
* type: тип отправителя: ИП, фабрика, склад, физ. лицо
* contact\_phone: телефон
* payment\_details: реквизиты для оплаты

*Связи:*

1 → M Order, один клиент, много заказов

* **Consignee**

*Описание:*

Получатель заказа. Также имеет доступ к отслеживанию груза. Может быть фабрикой, складским центром, просто ИП.

*Поля:*

* id: PK
* name: название компании
* type: тип отправителя: ИП, фабрика, склад, физ. лицо
* contact\_phone: телефон

*Связи:*

1 → M Order, один клиент, много заказов

* **Order**

*Описание:*

Сама перевозка. Может включать в себя много разных предметов перевозки. Имеет одного отправителя (customer) и одного получателя (shipper).

*Поля:*

* id: PK
* customer\_id: FK
* consignee\_id: FK
* created\_at: время создания
* planned\_pickup\_date: приблизительное время загрузки
* planned\_delivery\_date: приблизительное время отгрузки
* total\_price: общая цена
* route\_id: FK на весь маршрут
* current\_status\_id: FK на настоящий статус заказа. Имеется many-to-many таблица со всеми статусами заказа.
* order\_chat\_id

*Связи:*

1 → 1 Consignee (один заказ один получатель)

1 → 1 Customer (один заказ один отправитель)

1 → 1 Route (один заказ один маршрут)

1 → M Status (один заказ один текущий статус, но отслеживается история всех статусов в промежуточной many-to-many таблице)

1 → 1 Waybill (один заказ одна накладная)

1 → M Car (один заказ может иметь много машин)

M → M OrderCargo (много заказов имеют множество вещей, промежуточная many-to-many таблица)

1 → M OrderChat, у каждого заказа один чат, в котором много сообщений.

* **Waybill**

*Описание:*

Товарная накладная

*Поля:*

* id: PK
* customer\_send\_url: ссылка на подписание об отправке
* consignee\_accepted\_url: ссылка на подписание о принятии

*Связи:*

1 → 1 Order (одна накладная один заказ)

* **Car**

*Описание:*

Информация о грузовой машине.

*Поля:*

* id: PK
* max\_weight
* type
* plate\_number
* status\_id
* model
* production\_year
* last\_tech\_inspection\_date
* driver\_id

*Связи:*

Через промежуточную таблицу связывается с Order

1 → 1 Driver, на одну машину один водитель

* **Driver**

*Описание:*

Описание водителя

*Поля:*

* id: PK
* full\_name
* email
* hire\_date
* driver\_license\_number
* photo\_url
* status\_id
* gps\_tracker\_url
* rating

*Связи:*

1 → 1 DriverStatus, только один статус у водителя (свободен, в рейсе, в отпуске, уволен)

* **Cargo**

*Описание:*

Перевозимая единичная вещь, из которых состоит Order. Имеет свою степень хрупкости, прочие классификации.

*Поля:*

* id: PK
* name
* description
* weight\_kg
* volume\_m3
* cargo\_type\_id

*Связи:*

M → M OrderCargo, много заказов имеют множество вещей, промежуточная many-to-many таблица

1 → 1 CargoType, таблица с классификацией грузов по хрупкости.

* **Route**

*Описание:*

Маршрут перевозки с точками погрузки, разгрузки и промежуточными остановками. Интегрируется с Яндекс.Картами, но ключевые данные хранятся в бд.

*Поля:*

* id PK
* order\_id
* start\_point\_id
* end\_point\_id
* total\_distance\_km
* estimated\_duration\_min
* real\_start\_time
* real\_end\_time

*Связи:*

1→ 1 RoutePoint, точка старта

1 → 1 RoutePoint, точка финиша

M → M, у маршрута могут быть прочие места остановки.

* **RoutePoint**

*Описание:*

Хранит адреса, координаты и тип точки (погрузка, разгрузка, остановка)

*Поля:*

* id PK
* type\_id
* full\_address
* map\_mark\_url

*Связи:*

M → M, у маршрута могут быть прочие места остановки

1 → M, у точки может быть несколько статусов

В каждой таблице настроены индексы, что ускоряет поиск записей в таблице.

## Бизнес-функции

## \*Яшина Ева, Могушкова Фатима, Хисамудинова Эмилия

### 1. Управление заказами на грузоперевозки

## - Клиент указывает параметры груза (тип, вес, габариты), точки отправления и доставки, желаемые сроки.

## - Система анализирует доступных перевозчиков, их тарифы, рейтинги и предлагает оптимальные варианты.

- Для клиентов: Создание заказов, отслеживание груза, чат с перевозчиком.

### 2. Автоматизация и хранение документооборота

## - Система формирует типовые договоры перевозки, акты приёма-передачи груза.

## - Все документы хранятся в облаке и доступны в любой момент.

### 3. Контроль и мониторинг перевозок

* Если происходит внештатная ситуация, водитель отмечает в приложении о природе ситуации и уведомление поступает клиенту(чат с клиентом)

## - Интеграция с GPS-трекерами транспорта для отслеживания местоположения в реальном времени.

## - Автоматические оповещения о задержках, изменении маршрута, прибытии груза.

## - Формирование аналитики о выполненных перевозках (время в пути, расход топлива и т.д.).

- Приём заказов, навигация, отметки о погрузке/разгрузке.

### 4. Аналитика и оптимизация логистики

## - Автоматический расчёт цены за км с учётом типа груза, сезонности, спроса.

## -Оценка перевозчиков клиентами (надёжность, соблюдение сроков).

- Учёт характеристик машин, база данных с ними (грузоподъёмность, доступность).

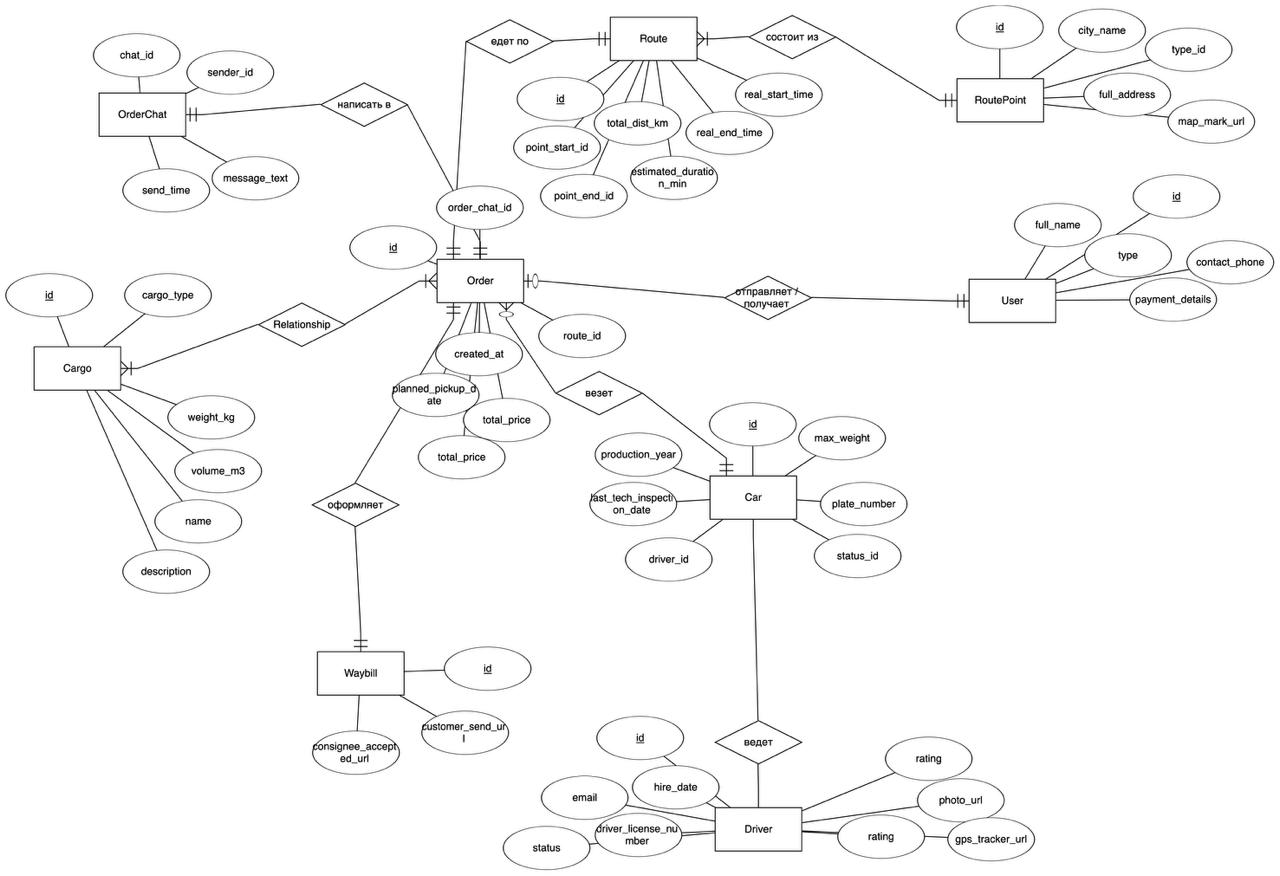
-Расчет топлива и его расхода

-Прибыль от конкретной машины

- Измерение отклонения от предполагаемого времени

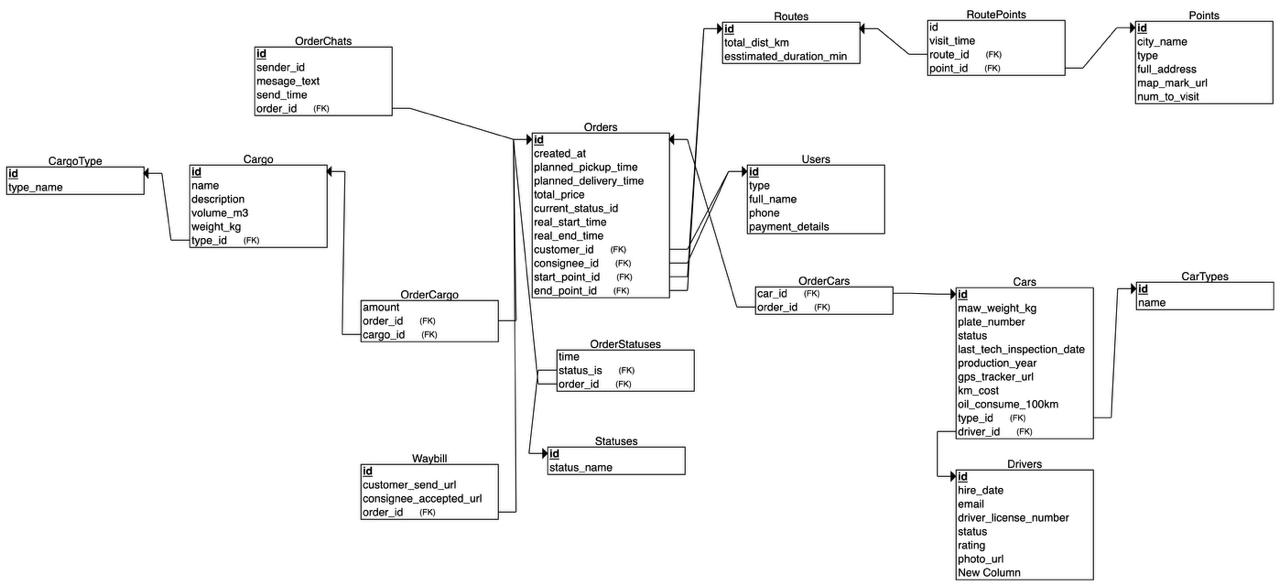
- Анализ загруженности автопарка

## Инфологическая



## 

## Даталогическая



## Третья нормальная форма (3NF) - обоснование

На данном этапе проверено отсутствие транзитивных зависимостей(когда неключевой атрибут зависит от другого неключевого атрибута). Соблюдение 3NF позволило создать гибкую, масштабируемую структуру БД, которая отвечает требованиям бизнес-логики в сфере логистики и грузоперевозок.

В нашей модели:

* Drivers содержит внешний ключ user\_id, ссылающийся на таблицу Users, где хранятся личные данные. Таким образом, информация не дублируется, а логически правильно разнесена между сущностями.
* В Cars данные о водителе (driver\_id) и технические параметры автомобиля напрямую зависят от идентификатора машины, и не зависят транзитивно от других неключевых атрибутов.
* Справочные данные вынесены в отдельные сущности: Statuses, CargoType, CarTypes. Это исключает дублирование текстовых описаний и поддерживает согласованность данных.

Таким образом, для каждой сущности в базе:

* Все неключевые атрибуты зависят **только от первичного ключа**,
* И не зависят от других неключевых атрибутов (отсутствуют транзитивные зависимости).

**Вывод**: База данных приведена к 3NF, что обеспечивает логическую целостность, устранение избыточности и отсутствие аномалий при вставке, обновлении или удалении данных.

Запросы к базе данных

1) выбрать все имена юзеров, которые являются отправителями (customers)

SELECT

full\_name AS Имя,

phone AS Телефон

FROM Users

WHERE type = 'customer';

2) выбрать города, где есть фабрики

SELECT

DISTINCT city\_name

FROM points

WHERE type = 'factory';

3) выбрать пользователей, которые отправляли сообщения

SELECT

DISTINCT full\_name

FROM (

SELECT full\_name

FROM orderchats och LEFT JOIN users u ON u.id = och.sender\_id

) AS user\_names;

4) выбрать сколько заказов СЕЙЧАС с каким статусом

SELECT

COUNT(o.id) AS Amount,

s.status\_name AS Status

FROM

orders o

LEFT JOIN statuses s ON o.current\_status\_id = s.id

GROUP BY s.id, s.status\_name

ORDER BY Status;

5) подсчитать сколько заказов у отправителей со статусом "в поездке"

SELECT

u.full\_name,

COUNT(o.id) AS Order\_Count

FROM (

SELECT

customer\_id, id

FROM orders WHERE current\_status\_id = (

SELECT id

FROM Statuses s

WHERE s.status\_name = 'In Transit'

)

) AS o

JOIN

users u ON u.id = o.customer\_id

GROUP BY u.full\_name

ORDER BY Order\_Count DESC;

6) анализ того, сколько в нашем автопарке машин

SELECT

name,

COUNT(\*) AS count

FROM (

SELECT name, cars.id

FROM

cars

LEFT JOIN cartypes ON cars.type\_id = cartypes.id

) as all\_cars

GROUP BY name

ORDER BY count;

7) типы машин, средняя стоимость грузов которых выше общего среднего среди грузов

SELECT

cartypes.name,

SUM(orders.total\_price) AS total\_amount

FROM orders

LEFT JOIN ordercars ON orders.id = ordercars.order\_id

LEFT JOIN cars ON cars.id = ordercars.car\_id

LEFT JOIN cartypes ON cartypes.id = cars.type\_id

GROUP BY cartypes.name

HAVING SUM(orders.total\_price) > (SELECT AVG(type\_price) FROM (

SELECT SUM(total\_price) AS type\_price

FROM orders

LEFT JOIN ordercars ON ordercars.car\_id = orders.id

LEFT JOIN cars ON ordercars.car\_id = cars.id

GROUP BY cars.type\_id

) as q

);

8) найти пользователей с самыми дорогими заказами

SELECT

full\_name,

phone

FROM users

WHERE EXISTS (

SELECT 52

FROM orders

WHERE orders.customer\_id = users.id AND orders.total\_price > (SELECT 0.99 \* MAX(total\_price) FROM orders)

);

9) сколько раз в наших маршрутах заезжали в разные города

SELECT

city\_name,

COUNT(\*) AS count\_visits

FROM

routepoints rp

LEFT JOIN routes r ON rp.route\_id = r.id

LEFT JOIN points p ON p.id = rp.point\_id

GROUP BY city\_name;

10) сколько в маршрутах остановок и сколько маршрутов с таким типом остановок

SELECT

num\_of\_places,

COUNT(\*) as routes\_with\_n\_stops

FROM

(

SELECT

COUNT(\*) AS num\_of\_places,

r.id

FROM

routepoints rp

LEFT JOIN routes r ON rp.route\_id = r.id

LEFT JOIN points p ON p.id = rp.point\_id

GROUP BY r.id

) as q

GROUP BY num\_of\_places;

11) среднее время в дороге в зависимости от количества остановок

SELECT

num\_of\_stops,

AVG(estimated\_duration\_min) AS medium\_roud\_time

FROM

(

SELECT

COUNT(\*) AS num\_of\_stops,

r.id AS route\_id,

AVG(r.estimated\_duration\_min) AS estimated\_duration\_min

FROM

routepoints rp

LEFT JOIN routes r ON rp.route\_id = r.id

LEFT JOIN points p ON p.id = rp.point\_id

GROUP BY r.id

) as q

GROUP BY num\_of\_stops;

12) для каждого маршрута и его точек подсчитать время проезда от одной точки до другой, а также подсчет суммарного времени

WITH RouteVisits AS (

SELECT

route\_id,

point\_id,

visit\_time,

DENSE\_RANK() OVER(

PARTITION BY route\_id

ORDER BY visit\_time

) AS number\_of\_visit,

ISNULL(DATEDIFF(minute, LAG(visit\_time) OVER (

PARTITION BY rp.route\_id

ORDER BY visit\_time

), visit\_time), 0) AS diff\_minutes

FROM

routepoints rp

LEFT JOIN routes r ON rp.route\_id = r.id

LEFT JOIN points p ON p.id = rp.point\_id

)

SELECT

route\_id,

point\_id,

number\_of\_visit,

diff\_minutes,

ISNULL(diff\_minutes + LAG(diff\_minutes) OVER (

PARTITION BY route\_id

ORDER BY visit\_time

), 0) AS total\_time

FROM RouteVisits;

13) время, проведенное в статусе у заказа

WITH OrderHistory AS (

SELECT

status\_name,

order\_id,

time

FROM

orderstatuses os

LEFT JOIN statuses s ON s.id = os.status\_id

LEFT JOIN orders o ON o.id = os.order\_id

)

SELECT

status\_name,

order\_id,

DENSE\_RANK() OVER (

PARTITION BY order\_id

ORDER BY time

) AS status\_seq\_number,

ISNULL(DATEDIFF(minute, time, LEAD(time) OVER(

PARTITION BY order\_id

ORDER BY time

)), 0) munutes\_in\_status

FROM OrderHistory;

15) проверим среднюю цену в Москве, Питере и остальных городах

SELECT

City,

ROUND(AVG(total\_price), 2)

FROM

(

SELECT

CASE

WHEN city\_name = 'Moscow' THEN 'MSK'

WHEN city\_name = 'Saint Petersburg' THEN 'SPB'

ELSE 'Others'

END AS City,

total\_price

FROM

Routes r

LEFT JOIN RoutePoints rp ON r.id = rp.route\_id

LEFT JOIN Points p ON p.id = rp.point\_id

LEFT JOIN Orders o ON o.id = r.order\_id

) as q1

GROUP BY City

16) выбрать города, в которых есть и заправка, и фабрика

SELECT

city\_name

FROM

points

WHERE

type = 'gas station'

INTERSECT

SELECT

city\_name

FROM

points

WHERE

type = 'factory';

17) найти пользователей, у которых нет заказа

SELECT

u.full\_name AS Customer\_Name,

u.phone AS Customer\_Phone

FROM

users u

LEFT JOIN

orders o ON u.id = o.customer\_id

WHERE

o.id IS NULL;

18) среднее потребление топлива у машины "Shacman X3000"

SELECT

AVG(c.oil\_consume\_100km) AS avg\_oil,

'Shacman X3000' AS car\_type

FROM Cars c LEFT JOIN CarTypes ct ON c.type\_id = ct.id

WHERE ct.id NOT IN (

SELECT id FROM CarTypes WHERE name = 'Shacman X3000'

);

19) выбрать типы вещей самые тяжелые типы вещей для грузоперевозок

SELECT

type\_name

FROM CargoType ct

WHERE ct.id IN (

SELECT cargotype.id

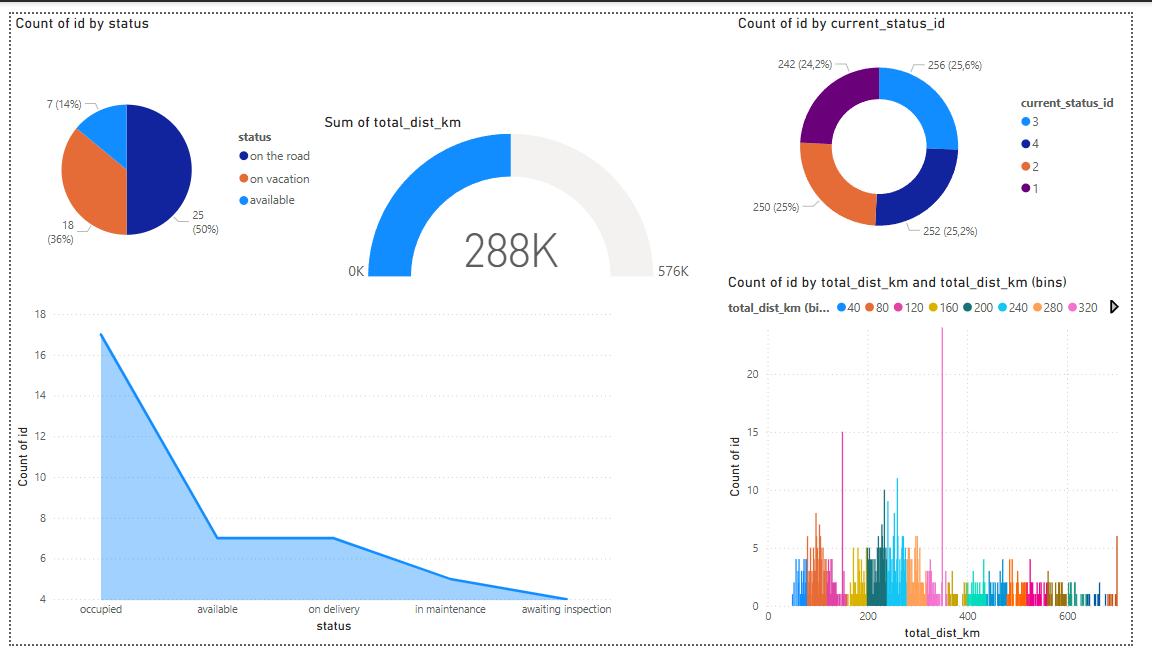
FROM cargo LEFT JOIN cargotype ON cargo.type\_id = cargotype.id

WHERE weight\_kg > (SELECT 0.5 \* MAX(weight\_kg) FROM cargo)

)

## Дашборды

Основной отчёт – агрегированные данные по широкому фильтру;



1)Диаграмма статусов водителей компании(левый верхний)

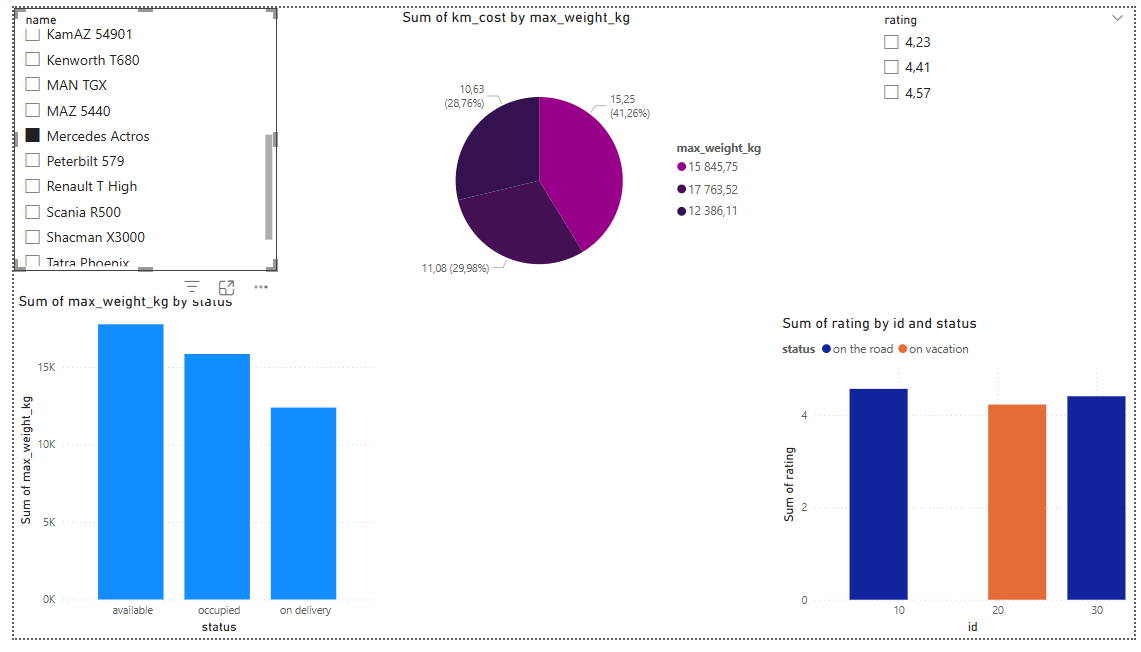
2)Диаграмма количества машин в зависимости от их статуса(левый нижний угол)

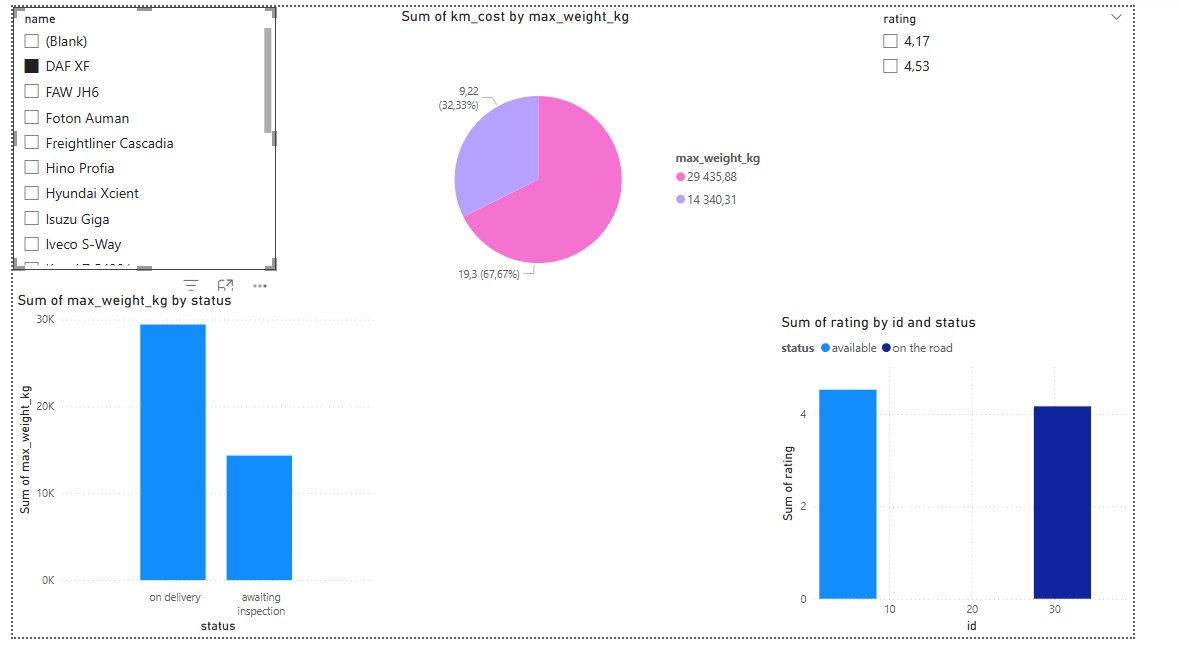
2)Диаграмма расстояния которое преодалела компания(верхняя центральная)

3)Диаграмма количества заказов в зависимости от их статуса(правый верхний угол)

4)диаграмма распределения расстояния при выполнении всех заказов(правый нижний угол)

* Детальный отчёт - по определенному узкому фильтру;





узкий фильтр - по моделям машин,но можно также срезать и по рейтингу водителей которые передвигаются на этих машинах

1)диаграмма стоимости километра в зависимости от грузоподъемности(центр верхний)

2)диаграмма грузоподъемности в зависимости от статуса машин(левый нижний угол)

3)диаграмма рейтинга наших водителей

|  |  |
| --- | --- |
|  |  |