Наталия Паламарчук (г.Усть-Каменогорск)

[nataliya.palamarchuk@kazzinc.com](mailto:nataliya.palamarchuk@kazzinc.com)

Аннотация

Данная база данных представляет собой комплексную систему управления гостиничным бизнесом, предназначенную для автоматизации учета номерного фонда, управления бронированиями и регистрации фактического проживания гостей.  
Она обеспечивает сквозной контроль финансовых операций, включая гибкую тарификацию услуг, учет дополнительных сервисов и фиксацию платежей различными способами.

уПРАВЛЕНИЕ БРОНИРОВАНИЕМ УСЛУГ ГОСТИНИЦЫ

Функционал для аналитиков

Оглавление

[1. Сущности 3](#_Toc221114860)

[Вспомогательные справочники: 3](#_Toc221114861)

[1. Статусы номеров (prj\_room\_statuses) 3](#_Toc221114862)

[2. Способы оплаты (prj\_payment\_methods) 3](#_Toc221114863)

[Раздел «Отели»: 3](#_Toc221114864)

[1. Отели (prj\_hotels) 3](#_Toc221114865)

[2. Категории номеров (prj\_room\_types) 3](#_Toc221114866)

[3. Ценовая сетка (prj\_hotel\_room\_prices) 3](#_Toc221114867)

[Номера (prj\_rooms) 3](#_Toc221114868)

[Гости (prj\_guests) 4](#_Toc221114869)

[Бронирование/заявки на бронь (prj\_reservations) 4](#_Toc221114870)

[Журнал фактического пребывания (prj\_check\_info) 4](#_Toc221114871)

[Раздел «Дополнительные услуги и Оплата» 4](#_Toc221114872)

[1. Каталог услуг (prj\_services) 4](#_Toc221114873)

[2. Потребленные услуги (prj\_guest\_services) 4](#_Toc221114874)

[3. Оплаты (prj\_payments) 5](#_Toc221114875)

[2. Связи 5](#_Toc221114876)

[1. Блок номерного фонда и цен 5](#_Toc221114877)

[2. Блок бронирования и регистрации 5](#_Toc221114878)

[3. Блок услуг и финансов 6](#_Toc221114879)

[3. ER-диаграмма 6](#_Toc221114880)

[4. Триггеры 7](#_Toc221114881)

[1. Пересечение бронирований 7](#_Toc221114882)

[2. Автоматическое изменение статуса номера 8](#_Toc221114883)

[5. Аналитические запросы 10](#_Toc221114884)

[1. Анализ доходности отелей 10](#_Toc221114885)

[2. Популярность доп. услуг по категориям 10](#_Toc221114886)

[3. Рейтинг самых прибыльных гостей 12](#_Toc221114887)

[4. Отчет по «плану-факту» проживания 13](#_Toc221114888)

[5. Поиск самых востребованных типов номеров в каждом отеле 14](#_Toc221114889)

[6. Сверка задолженностей 15](#_Toc221114890)

[7. Представления (Views) 18](#_Toc221114891)

[1. Cписок услуг по гостиницам в разрезе гостей 18](#_Toc221114892)

[2. Агрегированные услуги по гостю и гостинице 19](#_Toc221114893)

[3. Доля услуг в общем чеке гостя 21](#_Toc221114894)

[4. ТОП-3 услуги по каждой гостинице 23](#_Toc221114895)

[5. Средний чек услуг на гостя в гостинице 24](#_Toc221114896)

[8. Отчеты 25](#_Toc221114897)

[1. Оплаты услуг по гостиницам в разрезе гостей 25](#_Toc221114898)

[2. Гео-аналитика 26](#_Toc221114899)

[3. Комплексный Анализ доходности и эффективности 28](#_Toc221114900)

[9. Агрегация и сегментация данных 31](#_Toc221114901)

[1. RFM-анализ 31](#_Toc221114902)

[2. Анализ популярности и доходности услуг 34](#_Toc221114903)

[3. Сегментация по загрузке отелей 35](#_Toc221114904)

**Управления бронированием услуг гостиницы**

(***Функционал для аналитиков)***

# Сущности

## Вспомогательные справочники:

### Статусы номеров (prj\_room\_statuses)

* **status\_id**: Уникальный код статуса.
* **status\_name**: Текстовое описание (например, «Свободен», «Занят», «Уборка», «Ремонт» и т.д.).

### Способы оплаты (prj\_payment\_methods)

* **payment\_method\_id**: ID способа.
* **method\_name**: Название (например, "Наличные", "Карта", "Перевод" и т.д.).

## Раздел «Отели»:

### Отели (prj\_hotels)

* **hotel\_id**: Уникальный ключ отеля.
* **hotel\_name**: Название отеля.
* **address / phone / email**: Контактная информация.
* **stars**: Звездность. Ограничение CHECK гарантирует диапазон от 1 до 5.
* **total\_rooms**: Общее кол-во номеров (для статистики). Должно быть больше 0.
* **location**: Географические координаты (тип GEOGRAPHY). Позволяет вычислять расстояние до отеля или искать отели в радиусе на карте.

### Категории номеров (prj\_room\_types)

* **room\_type\_id**: ID категории.
* **type\_name**: Категория (например, "Комфорт", "Полулюкс", "Люкс" и т.д.).
* **capacity**: Вместимость (сколько человек может жить в номере).

### Ценовая сетка (prj\_hotel\_room\_prices)

* **hotel\_id / room\_type\_id**: Составной первичный ключ. Связывает конкретный отель с типом номера.
* **price**: Стоимость за сутки. Ограничение: цена всегда должна быть положительной.

## Номера (prj\_rooms)

* **room\_id**: Уникальный ID номера. Важное ограничение — нельзя создать два номера "10" в одном и том же отеле.
* **hotel\_id**: В каком отеле находится.
* **room\_type\_id**: К какому классу относится.
* **room\_number**: Номер на двери (строка, так как бывают номера "101а").
* **floor**: Этаж.
* **status\_id**: Текущее состояние (из справочника статусов).

## Гости (prj\_guests)

* **guest\_id**: Уникальный ID гостя.
* **first\_name / last\_name**: Имя и фамилия.
* **passport\_number**: Номер паспорта. Должен быть уникальным.
* **phone / email**: Контакты для связи и рассылки чеков.

## Бронирование/заявки на бронь (prj\_reservations)

* **reservation\_id**: Номер брони.
* **hotel\_id / room\_id / guest\_id**: Ссылки на то, кто, где и какой номер забронировал.
* **check\_in\_plan / check\_out\_plan**: Даты заезда и выезда по плану. Дата выезда не может быть раньше или равна дате заезда.
* **reservation\_date**: Дата создания брони (по умолчанию — текущий момент).

## Журнал фактического пребывания (prj\_check\_info)

* **check\_in\_id**: ID записи о регистрации.
* **reservation\_id**: Связь с бронью. По одной брони нельзя "заехать" дважды.
* **check\_in\_fact / check\_out\_fact**: Реальное время, когда гость переступил порог и когда сдал ключи.

## Раздел «Дополнительные услуги и Оплата»

### Каталог услуг (prj\_services)

* **service\_id**: ID услуги.
* **service\_name**: Наименование услуги (например, "Завтрак", "Бассейн", "Мини-бар", "СПА", и т.д.).
* **service\_price**: Стоимость единицы услуги.

### Потребленные услуги (prj\_guest\_services)

* **guest\_service\_id**: ID транзакции.
* **check\_in\_id**: К какому визиту гостя привязана услуга.
* **service\_id**: Какая именно услуга оказана.
* **service\_date**: Когда оказана.
* **quantity**: Количество (например, 2 завтрака). По умолчанию 1.

### Оплаты (prj\_payments)

* **payment\_id**: ID платежа.
* **reservation\_id**: К какому бронированию относится оплата.
* **payment\_date**: Когда заплатили.
* **amount**: Сумма платежа.
* **payment\_method\_id**: Как заплатили (нал/безнал).

Итого в системе предусмотрено **12** таблиц.

# 2. Связи

## 1. Блок номерного фонда и цен

* **Отели (prj\_hotels) → Номера (prj\_rooms) [1:N]**
  + **Связь:** Один отель содержит множество номеров.
* **Типы номеров (prj\_room\_types) → Номера (prj\_rooms) [1:N]**
  + **Связь:** Один тип (например, «Комфорт») может быть присвоен множеству физических комнат.
* **Статусы номеров (prj\_room\_statuses) → Номера (prj\_rooms) [1:N]**
  + **Связь:** Один статус (например, «Уборка») может одновременно относиться к нескольким номерам.
* **Отели (prj\_hotels) + Типы номеров (prj\_room\_types) → Цены в отелях (prj\_hotel\_room\_prices) [M:N]**
  + **Связь:** Многие ко многим. Позволяет устанавливать уникальную стоимость для конкретного типа номера в конкретном отеле.

## 2. Блок бронирования и регистрации

* **Гости (prj\_guests) → Бронирования (prj\_reservations) [1:N]**
  + **Связь:** Один гость может бронировать номера неограниченное количество раз за всё время.
* **Отели (prj\_hotels) → Бронирования (prj\_reservations) [1:N]**
  + **Связь:** В одном отеле оформляется множество бронирований.
* **Номера (prj\_rooms) → Бронирования (prj\_reservations) [1:N]**
  + **Связь:** Один и тот же номер может быть забронирован многократно (на разные временные периоды).
* **Бронирования (prj\_reservations) → Фактическое проживание (prj\_check\_info) [1:1]**
  + **Связь:** Один к одному. Одной записи о бронировании соответствует строго одна запись о реальном заезде/выезде гостя.

## 3. Блок услуг и финансов

* **Фактическое проживание (prj\_check\_info) → Услуги гостей (prj\_guest\_services) [1:N]**
  + **Связь:** За один период проживания гость может воспользоваться доп. услугами несколько раз.
* **Услуги (prj\_services) → Услуги гостей (prj\_guest\_services) [1:N]**
  + **Связь:** Одна услуга из каталога (например, «Завтрак») может быть оказана множеству разных гостей.
* **Бронирования (prj\_reservations) → Оплаты (prj\_payments) [1:N]**
  + **Связь:** Одно бронирование может включать несколько транзакций (например, предоплата 30% и доплата при выезде 70%).
* **Способы оплаты (prj\_payment\_methods) → Оплаты (prj\_payments) [1:N]**
  + **Связь:** Один способ (например, «Наличные») используется для оформления множества разных платежей.

Также следует отметить, что связь между prj\_reservations и prj\_check\_info должна быть реализована через UNIQUE ключ на reservation\_id. Это гарантирует, что система не даст создать две записи о заезде на одну и ту же бронь.

# 3. ER-диаграмма

Диаграмма, созданная средствами SQL Server Management Studio:

Изображение выглядит как текст, диаграмма, План, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

# Триггеры

## Пересечение бронирований

В системе следует предусмотреть триггер, который отслеживал бы пересечений бронирований. И делал запрет на бронирование, вызывая ошибку, если номер уже забронирован на указанный период:

CREATE OR ALTER TRIGGER trg\_no\_overbooking

ON prj\_reservations

--должен срабатывать и при вставке, и при изменении

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (

SELECT 1

FROM inserted i

JOIN prj\_reservations r ON r.room\_id = i.room\_id

WHERE

--проверка пересечения дат

i.check\_in\_plan < r.check\_out\_plan

AND i.check\_out\_plan > r.check\_in\_plan

--не сравнение запись с самой собой при UPDATE

AND i.reservation\_id <> r.reservation\_id

)

BEGIN

RAISERROR (N'Ошибка: Номер уже забронирован на указанный период или пересекается с существующей бронью.', 16, 1);

ROLLBACK TRANSACTION;

END

END;

Проверка триггера:

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Автоматическое изменение статуса номера

Когда гость заезжает (запись в prj\_check\_info), статус номера в таблице prj\_rooms должен автоматически меняться на «Занят».

CREATE OR ALTER TRIGGER trg\_AfterCheckIn

ON prj\_check\_info

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

UPDATE prj\_rooms

--'Занят'

SET status\_id = 2

FROM prj\_rooms r

JOIN prj\_reservations res ON r.room\_id = res.room\_id

JOIN inserted i ON res.reservation\_id = i.reservation\_id;

END;

Проверка триггера:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Статус изменился:  
Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

# Аналитические запросы

Сложные SQL-запросы для извлечения, агрегирования и анализа данных:

## Анализ доходности отелей

Этот запрос считает общую выручку каждого отеля, количество бронирований и средний чек. Используется **агрегация** данных и **Join**.

SELECT

    h.hotel\_name AS 'Отель',

    COUNT(r.reservation\_id) AS 'Всего бронирований',

    SUM(p.amount) AS 'Всего выручка',

    ROUND(AVG(p.amount), 2) AS 'Средний чек'

FROM prj\_hotels h

    JOIN prj\_reservations r ON h.hotel\_id = r.hotel\_id

    LEFT JOIN prj\_payments p ON r.reservation\_id = p.reservation\_id

GROUP BY h.hotel\_name

ORDER BY SUM(p.amount) DESC;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Популярность доп. услуг по категориям

В данном запросе используется **CTE**, чтобы сначала подготовить данные о продажах услуг, а затем объединить их с каталогом для финального анализа.

WITH ServiceStats AS (

    SELECT

        gs.service\_id,

        SUM(gs.quantity) AS total\_qty,

        --подсчет выручки

        SUM(gs.quantity \* s.service\_price) AS revenue\_per\_service

    FROM prj\_guest\_services gs

    --таблица с ценами

    JOIN prj\_services s ON gs.service\_id = s.service\_id

    GROUP BY gs.service\_id

)

SELECT

    s.service\_name AS 'Доп. услуга',

    ss.total\_qty AS 'Кол.-во заказов услуги',

    ss.revenue\_per\_service AS 'Выручка по услуге'

FROM prj\_services s

    JOIN ServiceStats ss ON s.service\_id = ss.service\_id

WHERE

    --возьмем только прибыльные услуги

    ss.revenue\_per\_service > 1000

ORDER BY ss.total\_qty DESC;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Рейтинг самых прибыльных гостей

Запрос использует оконную функцию **DENSE\_RANK()**, чтобы составить топ гостей по сумме всех их трат во всех отелях сети.

SELECT

    g.first\_name AS 'Имя',

    g.last\_name AS 'Фамилия',

    SUM(p.amount) AS 'Всего потрачено',

    --функция присваивает рейтинг по величине затраченной суммы, с большей суммой - рейтинг 1, и т.д.

    DENSE\_RANK() OVER (ORDER BY SUM(p.amount) DESC) AS 'Рейтинг гостя'

FROM prj\_guests g

    JOIN prj\_reservations r ON g.guest\_id = r.guest\_id

    JOIN prj\_payments p ON r.reservation\_id = p.reservation\_id

GROUP BY g.guest\_id, g.first\_name, g.last\_name

ORDER BY 4 ASC;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Отчет по «плану-факту» проживания

Запрос выявляет случаи, когда гости задерживались дольше запланированного или уезжали раньше, что важно для планирования уборки и брони.

SELECT

    g.last\_name + ' ' + g.first\_name AS 'Гость',

    r.check\_in\_plan AS 'Плановая дата заезда',

    r.check\_out\_plan AS 'Плановая дата выезда',

    ci.check\_in\_fact AS 'Фактическая дата заезда',

    ci.check\_out\_fact AS 'Фактическая дата выезда',

    DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan) AS 'Запланировано дней проживания',

    DATEDIFF(day, ci.check\_in\_fact, ci.check\_out\_fact) AS 'Фактически дней проживания',

    DATEDIFF(day, r.check\_out\_plan, ci.check\_out\_fact) AS 'Разница в днях',

    IIF(DATEDIFF(day, r.check\_out\_plan, ci.check\_out\_fact) > 0,

        'Клиент прожил дольше на ' + CAST(DATEDIFF(day, r.check\_out\_plan, ci.check\_out\_fact) AS NVARCHAR(3)) + ' дн.',

        'Клиент выехал раньше на ' + CAST(ABS(DATEDIFF(day, r.check\_out\_plan, ci.check\_out\_fact)) AS NVARCHAR(3)) + ' дн.')

    AS 'Статус'

FROM prj\_reservations r

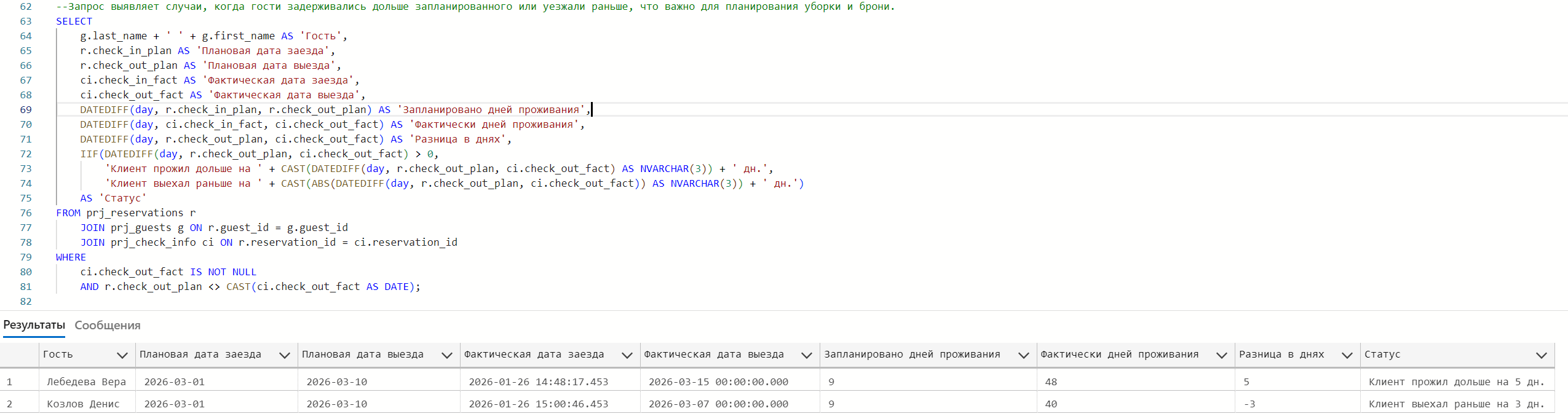
    JOIN prj\_guests g ON r.guest\_id = g.guest\_id

    JOIN prj\_check\_info ci ON r.reservation\_id = ci.reservation\_id

WHERE

    ci.check\_out\_fact IS NOT NULL

    AND r.check\_out\_plan <> CAST(ci.check\_out\_fact AS DATE);



## Поиск самых востребованных типов номеров в каждом отеле

Используется вложенный подзапрос для расчета процента загрузки конкретного типа номера относительно общего количества броней в отеле.

SELECT

    h.hotel\_name AS 'Отель',

    rt.type\_name AS 'Категория номера',

    COUNT(r.reservation\_id) AS 'Кол.-во бронирований',

    (SELECT COUNT(\*) FROM prj\_reservations WHERE hotel\_id = h.hotel\_id) AS 'Общее кол.-во броней',

    ROUND(CAST(COUNT(r.reservation\_id) AS FLOAT) /

        (SELECT COUNT(\*) FROM prj\_reservations WHERE hotel\_id = h.hotel\_id) \* 100, 2) AS '% загрузки'

FROM prj\_hotels h

    JOIN prj\_rooms rm ON h.hotel\_id = rm.hotel\_id

    JOIN prj\_room\_types rt ON rm.room\_type\_id = rt.room\_type\_id

    JOIN prj\_reservations r ON rm.room\_id = r.room\_id

GROUP BY h.hotel\_id, h.hotel\_name, rt.type\_name

ORDER BY 'Отель', 'Кол.-во бронирований' DESC;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Сверка задолженностей

Запрос находит бронирования, по которым сумма оказанных услуг и стоимость проживания превышают фактически внесенные платежи. Используется **Having** и математический расчет.

SELECT

    r.reservation\_id,

    g.guest\_id,

    g.last\_name + ' ' + g.first\_name AS 'Гость',

    hp.price AS 'Цена',

    DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan) AS 'План.проживание(дн.)',

    --стоимость номера

    (hp.price \* DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan)) AS 'План.стоимость номера',

    SUM(gs.quantity) AS 'Кол.-во оказ. услуг',

    ISNULL(SUM(gs.quantity \* s.service\_price), 0) AS 'Стоимость оказ.услуг',

    ISNULL(p.total\_paid, 0) AS 'Всего оплачено',

    ((hp.price \* DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan)) + ISNULL(SUM(gs.quantity \* s.service\_price), 0)) - ISNULL(p.total\_paid, 0) AS 'Задолженность'

FROM prj\_reservations r

    JOIN prj\_guests g ON r.guest\_id = g.guest\_id

    LEFT JOIN prj\_rooms room ON r.room\_id = room.room\_id

    JOIN prj\_hotel\_room\_prices hp ON r.hotel\_id = hp.hotel\_id AND room.room\_type\_id = hp.room\_type\_id

    LEFT JOIN prj\_check\_info ci ON r.reservation\_id = ci.reservation\_id

    LEFT JOIN prj\_guest\_services gs ON ci.check\_in\_id = gs.check\_in\_id

    LEFT JOIN prj\_services s ON gs.service\_id = s.service\_id

    LEFT JOIN

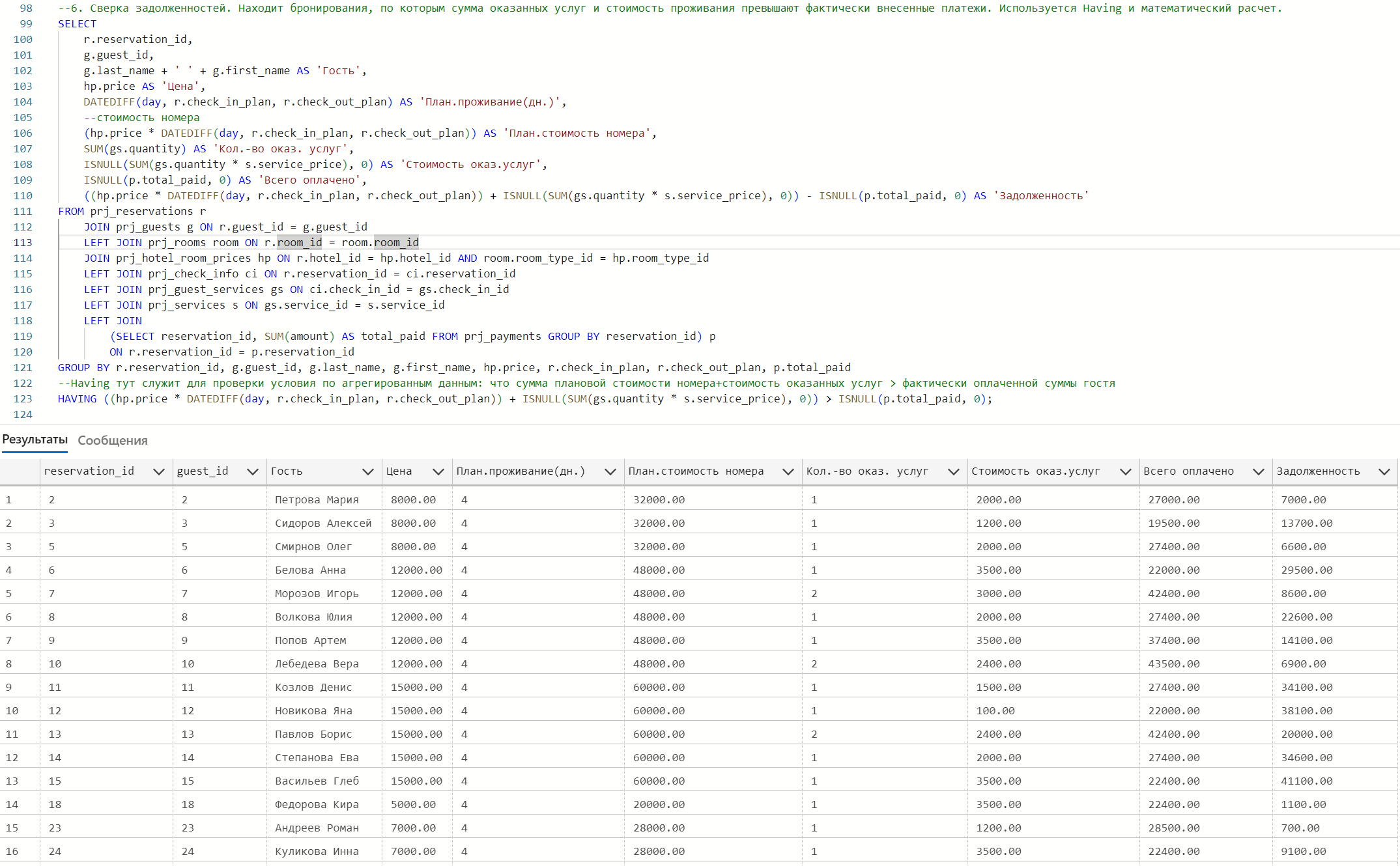
        (SELECT reservation\_id, SUM(amount) AS total\_paid FROM prj\_payments GROUP BY reservation\_id) p

        ON r.reservation\_id = p.reservation\_id

GROUP BY r.reservation\_id, g.guest\_id, g.last\_name, g.first\_name, hp.price, r.check\_in\_plan, r.check\_out\_plan, p.total\_paid

--Having тут служит для проверки условия по агрегированным данным: что сумма плановой стоимости номера+стоимость оказанных услуг > фактически оплаченной суммы гостя

HAVING ((hp.price \* DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan)) + ISNULL(SUM(gs.quantity \* s.service\_price), 0)) > ISNULL(p.total\_paid, 0);



Этот запрос можно оптимизировать с использованием двух CTE:

--суммы по услугам

WITH ServiceTotals AS (

    SELECT

        ci.reservation\_id,

        SUM(gs.quantity) AS total\_qty,

        SUM(gs.quantity \* s.service\_price) AS total\_service\_price

    FROM prj\_check\_info ci

    JOIN prj\_guest\_services gs ON ci.check\_in\_id = gs.check\_in\_id

    JOIN prj\_services s ON gs.service\_id = s.service\_id

    GROUP BY ci.reservation\_id

),

--платежи

PaymentTotals AS (

    SELECT

        reservation\_id,

        SUM(amount) AS total\_paid

    FROM prj\_payments

    GROUP BY reservation\_id

)

--теперь в основном запросе не обязательно использовать Having

SELECT

    r.reservation\_id,

    g.guest\_id,

    g.last\_name + ' ' + g.first\_name AS 'Гость',

    hp.price AS 'Цена',

    DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan) AS 'План.проживание(дн.)',

    (hp.price \* DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan)) AS 'План.стоимость номера',

    ISNULL(st.total\_qty, 0) AS 'Кол.-во оказ. услуг',

    ISNULL(st.total\_service\_price, 0) AS 'Стоимость оказ.услуг',

    ISNULL(pt.total\_paid, 0) AS 'Всего оплачено',

    ((hp.price \* DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan)) + ISNULL(st.total\_service\_price, 0)) - ISNULL(pt.total\_paid, 0) AS 'Задолженность'

FROM prj\_reservations r

    JOIN prj\_guests g ON r.guest\_id = g.guest\_id

    LEFT JOIN prj\_rooms room ON r.room\_id = room.room\_id

    JOIN prj\_hotel\_room\_prices hp ON r.hotel\_id = hp.hotel\_id AND room.room\_type\_id = hp.room\_type\_id

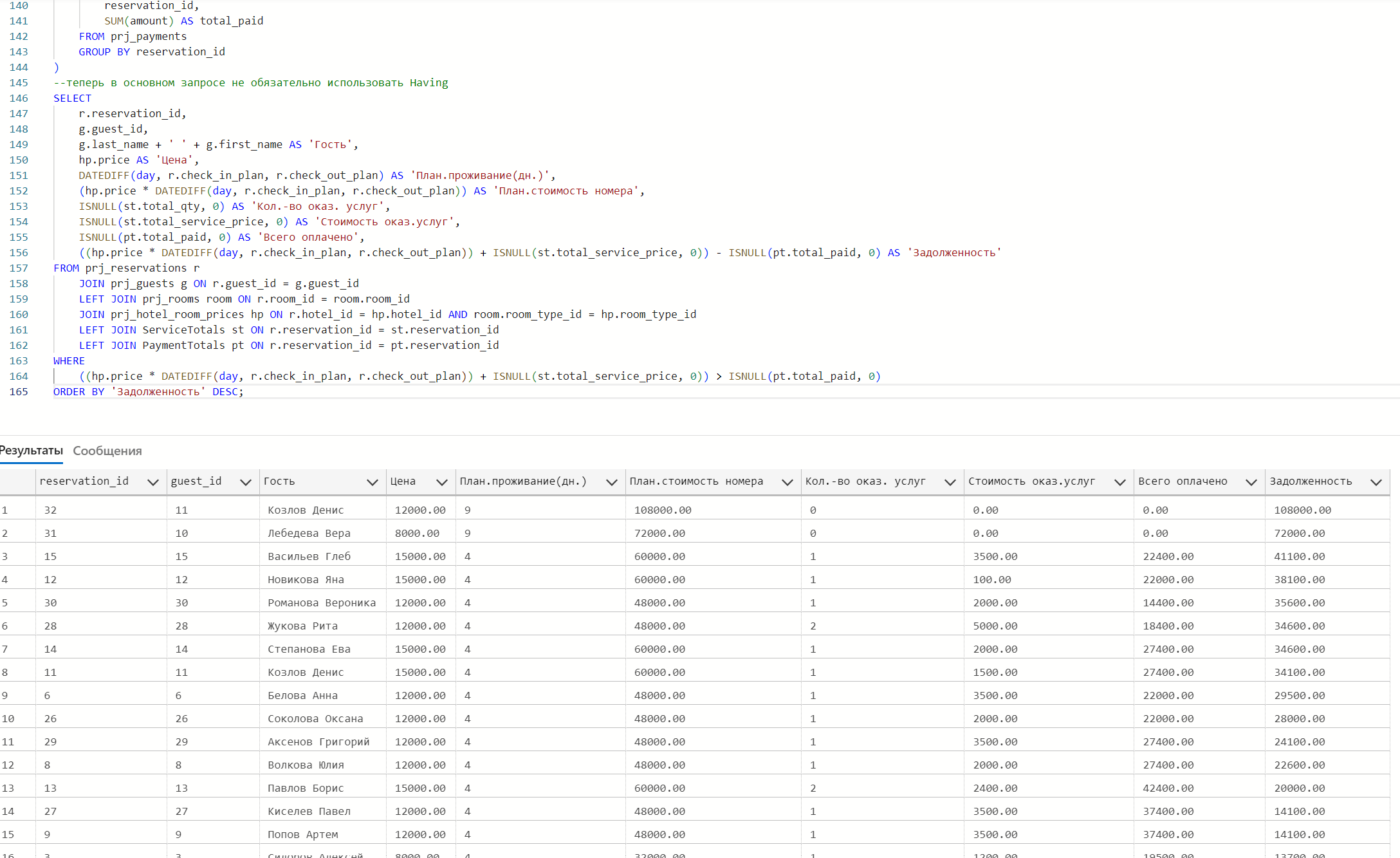
    LEFT JOIN ServiceTotals st ON r.reservation\_id = st.reservation\_id

    LEFT JOIN PaymentTotals pt ON r.reservation\_id = pt.reservation\_id

WHERE

    ((hp.price \* DATEDIFF(day, r.check\_in\_plan, r.check\_out\_plan)) + ISNULL(st.total\_service\_price, 0)) > ISNULL(pt.total\_paid, 0)

ORDER BY 'Задолженность' DESC;



# Представления (Views)

## Cписок услуг по гостиницам в разрезе гостей

Может применяться в отчётах по доп. услугам, детализация чеков гостей и т.д.

CREATE VIEW prj\_guest\_services\_by\_hotel

AS

SELECT

    h.hotel\_id,

    h.hotel\_name,

    g.guest\_id,

    CONCAT(g.last\_name, ' ', g.first\_name) AS guest\_name,

    s.service\_name,

    gs.service\_date,

    gs.quantity,

    s.service\_price,

    gs.quantity \* s.service\_price AS total\_amount

FROM prj\_guest\_services gs

    JOIN prj\_check\_info ci ON gs.check\_in\_id = ci.check\_in\_id

    JOIN prj\_reservations r ON ci.reservation\_id = r.reservation\_id

    JOIN prj\_hotels h ON r.hotel\_id = h.hotel\_id

    JOIN prj\_guests g ON r.guest\_id = g.guest\_id

    JOIN prj\_services s ON gs.service\_id = s.service\_id;

Изображение выглядит как текст, число, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Агрегированные услуги по гостю и гостинице

Сумма услуг одного гостя в одной гостинице. Реализовано с CTE. Применение: анализ популярности услуг.

CREATE VIEW prj\_guest\_service\_summary

AS

WITH ServiceCTE AS (

    SELECT

        h.hotel\_id,

        h.hotel\_name,

        g.guest\_id,

        CONCAT(g.last\_name, ' ', g.first\_name) AS guest\_name,

        s.service\_name,

        gs.quantity \* s.service\_price AS service\_amount

    FROM prj\_guest\_services gs

    JOIN prj\_check\_info ci ON gs.check\_in\_id = ci.check\_in\_id

    JOIN prj\_reservations r ON ci.reservation\_id = r.reservation\_id

    JOIN prj\_hotels h ON r.hotel\_id = h.hotel\_id

    JOIN prj\_guests g ON r.guest\_id = g.guest\_id

    JOIN prj\_services s ON gs.service\_id = s.service\_id

)

SELECT

    hotel\_name,

    guest\_name,

    service\_name,

    SUM(service\_amount) AS total\_service\_amount

FROM ServiceCTE

GROUP BY hotel\_name, guest\_name, service\_name;

Изображение выглядит как текст, снимок экрана, число, документ

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Доля услуг в общем чеке гостя

Используются оконные функции SUM() OVER (). Может применяться для анализа структуры расходов гостя, персональных предложений, маркетинговой аналитика

CREATE VIEW prj\_guest\_service\_share

AS

WITH TotalPerGuest AS (

    SELECT

        g.guest\_id,

        CONCAT(g.last\_name, ' ', g.first\_name) AS guest\_name,

        h.hotel\_name,

        s.service\_name,

        gs.quantity \* s.service\_price AS service\_amount,

        SUM(gs.quantity \* s.service\_price) OVER (

            PARTITION BY g.guest\_id, h.hotel\_id

        ) AS total\_guest\_services

    FROM prj\_guest\_services gs

    JOIN prj\_check\_info ci ON gs.check\_in\_id = ci.check\_in\_id

    JOIN prj\_reservations r ON ci.reservation\_id = r.reservation\_id

    JOIN prj\_hotels h ON r.hotel\_id = h.hotel\_id

    JOIN prj\_guests g ON r.guest\_id = g.guest\_id

    JOIN prj\_services s ON gs.service\_id = s.service\_id

)

SELECT

    hotel\_name,

    guest\_name,

    service\_name,

    service\_amount,

    total\_guest\_services,

    CAST(service\_amount \* 100.0 / total\_guest\_services AS DECIMAL(5,2)) AS service\_share\_percent

FROM TotalPerGuest;

Изображение выглядит как текст, число, снимок экрана, документ

Содержимое, созданное искусственным интеллектом, может быть неверным.

## ТОП-3 услуги по каждой гостинице

Используется оконная функция RANK. Применение: KPI гостиницы, оптимизация сервиса, отчёты руководству

CREATE VIEW prj\_top\_services\_by\_hotel

AS

WITH ServiceRank AS (

    SELECT

        h.hotel\_name,

        s.service\_name,

        SUM(gs.quantity \* s.service\_price) AS revenue,

        RANK() OVER (

            PARTITION BY h.hotel\_id

            ORDER BY SUM(gs.quantity \* s.service\_price) DESC

        ) AS service\_rank

    FROM prj\_guest\_services gs

    JOIN prj\_check\_info ci ON gs.check\_in\_id = ci.check\_in\_id

    JOIN prj\_reservations r ON ci.reservation\_id = r.reservation\_id

    JOIN prj\_hotels h ON r.hotel\_id = h.hotel\_id

    JOIN prj\_services s ON gs.service\_id = s.service\_id

    GROUP BY h.hotel\_id, h.hotel\_name, s.service\_name

)

SELECT \*

FROM ServiceRank

WHERE service\_rank <= 3;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Средний чек услуг на гостя в гостинице

CREATE VIEW prj\_avg\_service\_check\_by\_hotel

AS

WITH GuestTotals AS (

    SELECT

        h.hotel\_name,

        g.guest\_id,

        SUM(gs.quantity \* s.service\_price) AS guest\_service\_total

    FROM prj\_guest\_services gs

    JOIN prj\_check\_info ci ON gs.check\_in\_id = ci.check\_in\_id

    JOIN prj\_reservations r ON ci.reservation\_id = r.reservation\_id

    JOIN prj\_hotels h ON r.hotel\_id = h.hotel\_id

    JOIN prj\_guests g ON r.guest\_id = g.guest\_id

    JOIN prj\_services s ON gs.service\_id = s.service\_id

    GROUP BY h.hotel\_name, g.guest\_id

)

SELECT

    hotel\_name,

    AVG(guest\_service\_total) AS avg\_service\_check

FROM GuestTotals

GROUP BY hotel\_name;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

# Отчеты

## Оплаты услуг по гостиницам в разрезе гостей

Отчет будет создаваться в Power BI Desktop, на основе созданного ранее представления

[prj\_guest\_services\_by\_hotel](#_Cписок_услуг_по).

В отчете создана мера с общей суммой оплаты по отелю, созданы фильтры по отелю, гостю и услуге, а также выводятся 2 диаграммы, позволяющие просматривать сумму оплаты по датам, а также сумму оплаты в разрезе услуг. С помощью фильтров можно выбирать отель, гостя и услугу, графики динамически меняются, предоставляя информацию по требуемым параметрам.

На второй странице отчета размещена круговая диаграмма, показывающая долю услуги в общей выручке.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Гео-аналитика

Поскольку в таблице prj\_hotels есть поле location с типом GEOGRAPHY, можно сделать визуализацию данных на карте: размер «пузырька» отеля — это его доход, а цвет — процент загрузки.

Для такого отчета придется создать новое представление:

CREATE VIEW prj\_Hotel\_KPIs AS

WITH HotelKPIs AS (

    SELECT

        h.hotel\_id,

        h.hotel\_name,

        --преобразование GEOGRAPHY в читаемые координаты

        h.location.Lat AS Latitude,

        h.location.Long AS Longitude,

        h.total\_rooms,

        --доход (размер пузырька)

        SUM(p.amount) AS TotalRevenue,

        --загрузка (цвет пузырька)

        CAST(COUNT(DISTINCT ci.check\_in\_id) \* 100.0 / (h.total\_rooms \* 30) AS DECIMAL(10,2)) AS OccupancyRate

    FROM prj\_hotels h

    LEFT JOIN prj\_reservations r ON h.hotel\_id = r.hotel\_id

    LEFT JOIN prj\_check\_info ci ON r.reservation\_id = ci.reservation\_id

    LEFT JOIN prj\_payments p ON r.reservation\_id = p.reservation\_id

    --фильтр за последний месяц

    WHERE r.check\_in\_plan >= DATEADD(month, -1, GETDATE())

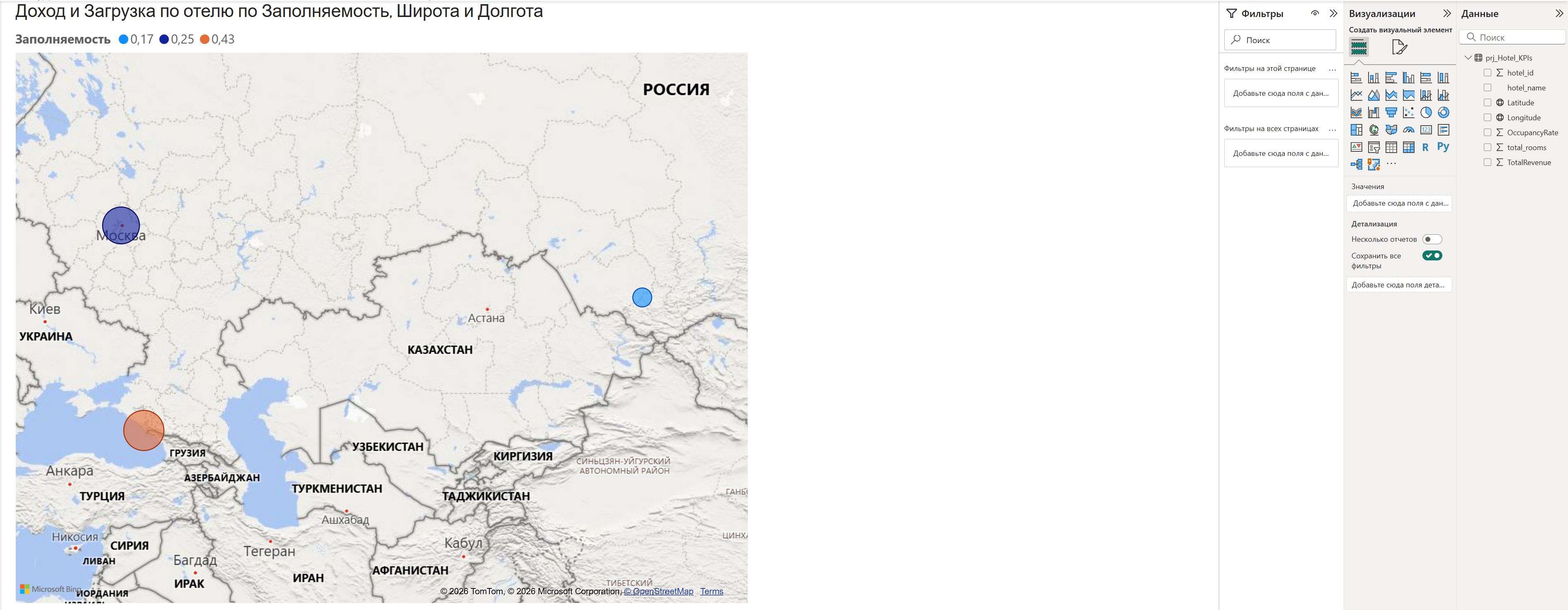
    GROUP BY h.hotel\_id, h.hotel\_name, h.location.Lat, h.location.Long, h.total\_rooms

)

SELECT \* FROM HotelKPIs;

Далее в Power BI Desktop на визуализации «Карта» можно вывести эти «пузырьки», настроив их цвет и размер. Цель такого отчета - подсветить «аномалии»:

* **Большой красный пузырь:** Отель в прибыльном месте, приносит много денег, но работает «вполсилы» (низкая загрузка).
* **Маленький зеленый пузырь:** Отель забит под завязку, но денег мало. Скорее всего, цены неоправданно занижены.



## Комплексный Анализ доходности и эффективности

Один из самых важных отчетов для гостиничного бизнеса — это «Комплексный анализ доходности и эффективности (RevPAR)». Он должен учитывать:

1. Финансовые показатели: Выручка за проживание и выручка за доп. услуги.
2. Эффективность (RevPAR): Доход на один доступный номер.
3. Лояльность: Лучший клиент отеля, кто из гостей потратил больше всего денег за всё время.
4. Операционная точность: Сравнение плановых заездов с фактическим количеством проданных ночей.
5. Средняя цена за ночь.

Используем хранимую процедуру для сбора данных по этому отчету:

ALTER PROCEDURE sp\_Hotel\_Analytics

    @StartDate DATE = NULL,

    @EndDate DATE = NULL,

    @HotelID NVARCHAR(50) = NULL

AS

-- DECLARE @StartDate DATE = NULL;

-- DECLARE @EndDate DATE = NULL;

-- DECLARE @HotelID NVARCHAR(50) = '1,2,3';

BEGIN

    SET NOCOUNT ON;

    --если даты не указаны, то текущий год

    SET @StartDate = ISNULL(@StartDate, DATEFROMPARTS(YEAR(GETDATE()), 1, 1));

    SET @EndDate = ISNULL(@EndDate, DATEFROMPARTS(YEAR(GETDATE()), 12, 31));

    WITH RoomRevenue AS (

        --доход от проживания (через платежи, привязанные к брони), должен брать только те платежи, что попали в период

        SELECT

            r.hotel\_id,

            r.reservation\_id,

            SUM(p.amount) as total\_paid

        FROM prj\_reservations r

            JOIN prj\_payments p ON r.reservation\_id = p.reservation\_id

        WHERE p.payment\_date BETWEEN @StartDate AND @EndDate

        GROUP BY r.hotel\_id, r.reservation\_id

    ),

    ServiceRevenue AS (

        --доход от доп. услуг, тоже за период

        SELECT

            res.hotel\_id,

            SUM(gs.quantity \* s.service\_price) as service\_total

        FROM prj\_guest\_services gs

            JOIN prj\_services s ON gs.service\_id = s.service\_id

            JOIN prj\_check\_info ci ON gs.check\_in\_id = ci.check\_in\_id

            JOIN prj\_reservations res ON ci.reservation\_id = res.reservation\_id

        WHERE gs.service\_date BETWEEN @StartDate AND @EndDate

        GROUP BY res.hotel\_id

    ),

    Occupancy AS (

        --количество фактически завершенных броней и ночей

        SELECT

            r.hotel\_id,

            COUNT(ci.reservation\_id) as total\_stays,

            SUM(DATEDIFF(day, ci.check\_in\_fact, ci.check\_out\_fact)) as total\_nights

        FROM prj\_reservations r

            JOIN prj\_check\_info ci ON r.reservation\_id = ci.reservation\_id

        WHERE ci.check\_in\_fact >= @StartDate AND ci.check\_out\_fact <= @EndDate

        GROUP BY hotel\_id

    )

    SELECT

        h.hotel\_name AS 'Отель',

        h.stars AS 'Звезд',

        occ.total\_stays AS 'Кол.-во бронирований',

        occ.total\_nights AS 'Продано ночей',

        --платежи, доходность

        ISNULL(rev.total\_room\_income, 0) AS 'Доход от номеров',

        ISNULL(srv.service\_total, 0) AS 'Доход от услуг',

        ISNULL(rev.total\_room\_income, 0) + ISNULL(srv.service\_total, 0) AS 'Общая выручка',

        --KPI (Key Performance Indicators) - Ключевые показатели эффективности

        CASE

            WHEN occ.total\_nights > 0 THEN CAST(ISNULL(rev.total\_room\_income, 0) / occ.total\_nights AS DECIMAL(10,2))

            ELSE 0

        --ADR (Average Daily Rate)

        END AS 'ADR (Средняя цена за ночь)',

        --эффективность использования фонда: RevPAR (Revenue Per Available Room) - доход на один доступный номер, учитывает и цену, и заполняемость

        --RevPAR = Общая выручка от номеров / Общее кол-во доступных номеров в периоде

        CAST((ISNULL(rev.total\_room\_income, 0) / (h.total\_rooms \* DATEDIFF(day, @StartDate, DATEADD(day, 1, @EndDate)))) AS DECIMAL(10,2)) AS 'RevPAR',

        --лучший гость

        (SELECT TOP 1 g.last\_name + ' ' + g.first\_name

         FROM prj\_guests g

            JOIN prj\_reservations r ON g.guest\_id = r.guest\_id

         WHERE r.hotel\_id = h.hotel\_id

         GROUP BY g.guest\_id, g.last\_name, g.first\_name

         ORDER BY COUNT(r.reservation\_id) DESC) AS 'Самый лояльный гость'

    FROM prj\_hotels h

        LEFT JOIN (SELECT hotel\_id, SUM(total\_paid) as total\_room\_income FROM RoomRevenue GROUP BY hotel\_id) rev ON h.hotel\_id = rev.hotel\_id

        LEFT JOIN ServiceRevenue srv ON h.hotel\_id = srv.hotel\_id

        LEFT JOIN Occupancy occ ON h.hotel\_id = occ.hotel\_id

    WHERE (@HotelID IS NULL OR h.hotel\_id = @HotelID)

    ORDER BY 'Общая выручка' DESC;

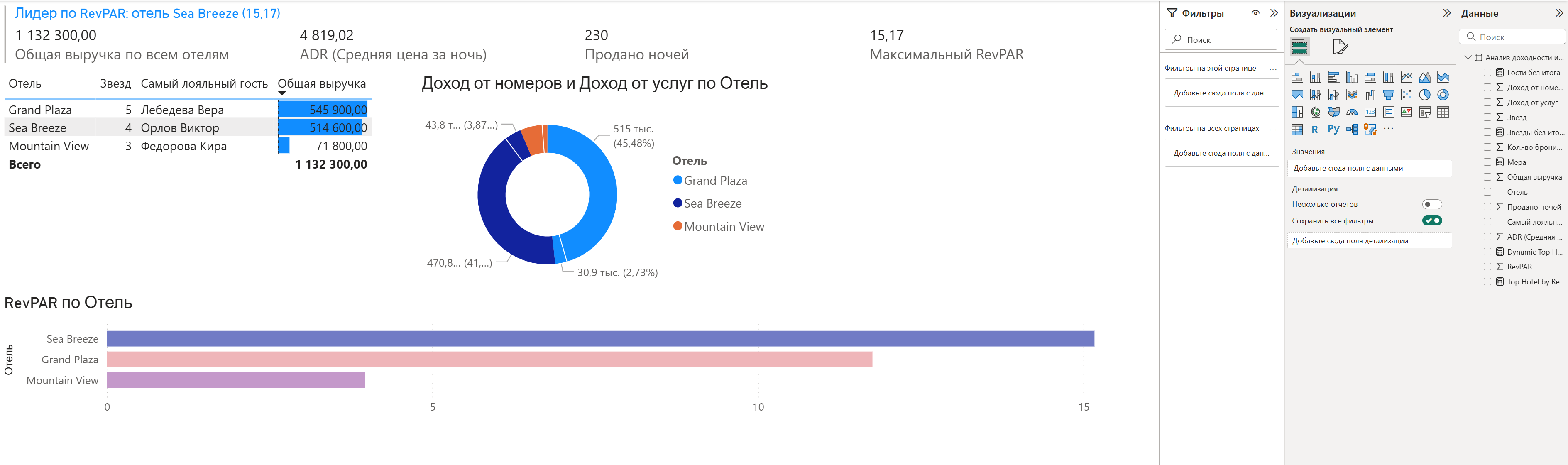
END;

Результат выполнения хранимой процедуры:

Изображение выглядит как текст, линия, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Отчет Power Bi по данным хранимой процедуры:



# Агрегация и сегментация данных

## RFM-анализ

Можно произвести сегментацию данных о гостях на основе RFM-анализа. **RFM-анализ** — это метод сегментации клиентов, который помогает понять, кто из них приносит больше всего денег, а кто может уйти к конкурентам.

Вместо того, чтобы применять ко всем гостям одинаковый подход, можно поделить базу на группы на основе их реального поведения. Название метода — это аббревиатура трех показателей:

1. **Recency (Давность):** Как давно гость останавливался в отеле? Чем меньше времени прошло, тем выше вероятность, что он вернется снова.
2. **Frequency (Частота):** Как часто гость останавливался в отеле? Помогает отличить случайных клиентов от преданных.
3. **Monetary (Деньги):** Сколько денег потратил гость? Показывает общую ценность клиента для бизнеса.

RFM-анализ можно представить примерно в таком виде:

SELECT

    g.last\_name AS 'Фамилия',

    g.first\_name AS 'Имя',

    --Recency (Давность) - от даты фактического заезда (check\_in\_fact)

    DATEDIFF(day, MAX(ci.check\_in\_fact), GETDATE()) AS 'Давность(дней)',

    --Frequency (Частота) - только те бронирования, по которым был реальный заезд

    COUNT(ci.check\_in\_id) AS 'Частота(визитов)',

    --Monetary (Деньги) - сумма платежей по факту проживания

    SUM(p.amount) AS 'Сумма',

    CASE

        --клиент был недавно и часто посещает

        WHEN COUNT(ci.check\_in\_id) >= 2 AND DATEDIFF(day, MAX(ci.check\_in\_fact), GETDATE()) <= 30 THEN 'Активный лояльный'

        --был один раз и очень давно (фактически заехал, но не вернулся)

        WHEN COUNT(ci.check\_in\_id) = 1 AND DATEDIFF(day, MAX(ci.check\_in\_fact), GETDATE()) > 30 THEN 'Разовый (ушедший)'

        --много визитов, но последний был больше полугода назад

        WHEN COUNT(ci.check\_in\_id) >= 3 AND DATEDIFF(day, MAX(ci.check\_in\_fact), GETDATE()) > 180 THEN 'Спящий VIP'

        --оставил много денег за последний визит

        WHEN SUM(p.amount) > 40000 AND DATEDIFF(day, MAX(ci.check\_in\_fact), GETDATE()) <= 30 THEN 'Ценный гость'

        ELSE 'Прочие'

    END AS 'RFM-сегмент(факт.)'

FROM prj\_guests g

    JOIN prj\_reservations r ON g.guest\_id = r.guest\_id

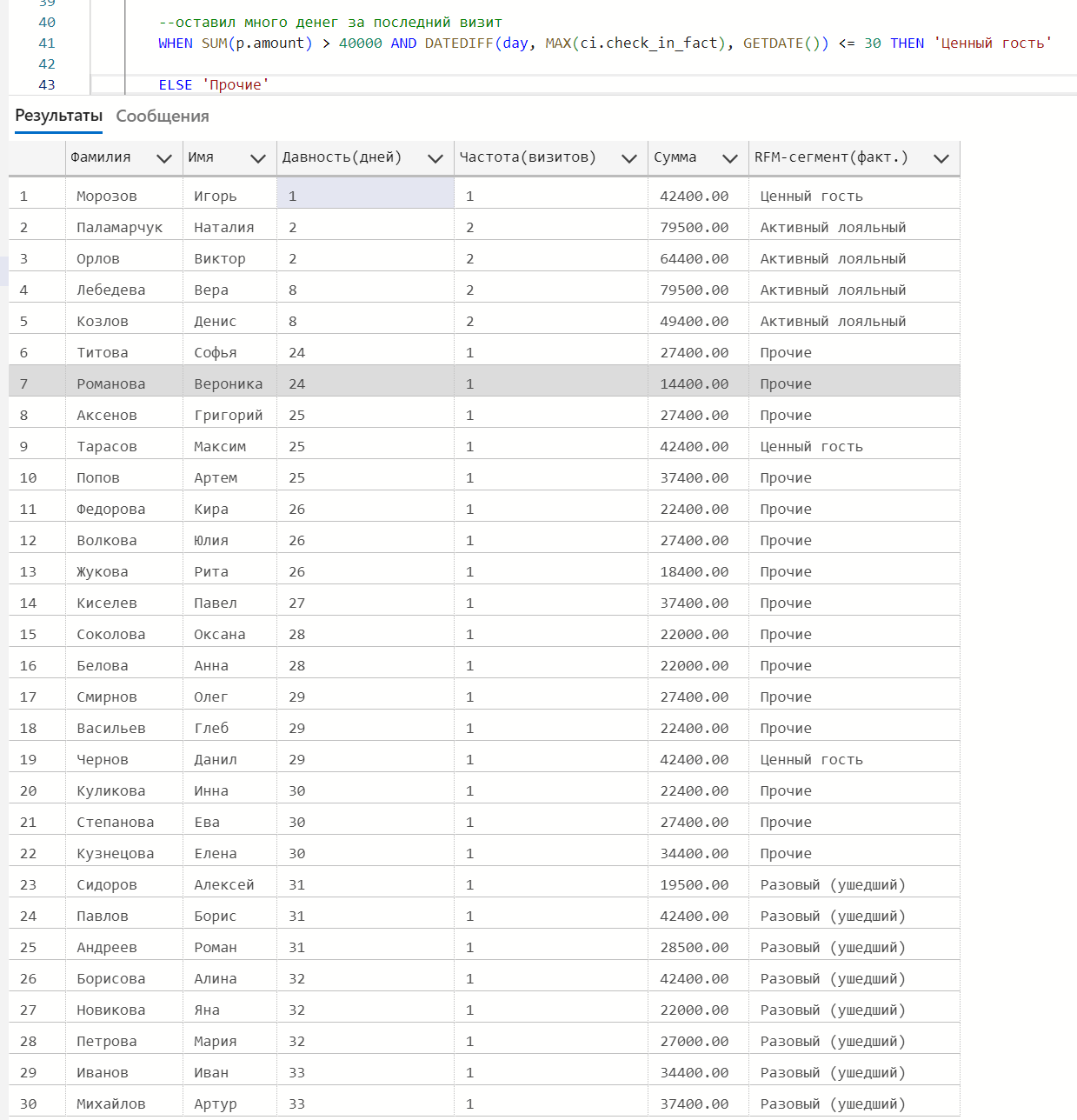
    --INNER JOIN, чтобы исключить брони без заезда

    INNER JOIN prj\_check\_info ci ON r.reservation\_id = ci.reservation\_id

    JOIN prj\_payments p ON r.reservation\_id = p.reservation\_id

GROUP BY g.guest\_id, g.last\_name, g.first\_name

ORDER BY 3 ASC;



## Анализ популярности и доходности услуг

Сегментирование дополнительных услуг, чтобы понять, какие из них приносят основной доход.

Число **0,2** в запросе — отсылка к закону Парето: «20% усилий дают 80% результата», или «20% видов услуг приносят 80% всей выручки».

Используя эту логику, можно выделить самые выгодные услуги:

1. Вычислить **общую выручку** всех проданных услуг отеля.
2. Умножить её на **0,2**, получая «планку» в 20% от общей выручки.
3. Сравнить выручку каждой конкретной услуги с этой планкой.

Таким образом, некоторые услуги получают статус **«Ключевой актив»**.

SELECT

    s.service\_name AS 'Услуга',

    SUM(gs.quantity) AS 'Кол.-во заказов',

    SUM(gs.quantity \* s.service\_price) AS 'Выручка',

    CASE

        --если выручка по услуге > 20% от общей выручки - закон Парето: «20% видов услуг приносят 80% всей выручки»

        WHEN SUM(gs.quantity \* s.service\_price) > (SELECT SUM(quantity \* service\_price) \* 0.2 FROM prj\_guest\_services JOIN prj\_services ON prj\_guest\_services.service\_id = prj\_services.service\_id) THEN 'Ключевой актив'

        ELSE 'Второстепенная услуга'

    END AS 'Категория'

FROM prj\_services s

LEFT JOIN prj\_guest\_services gs ON s.service\_id = gs.service\_id

GROUP BY s.service\_id, s.service\_name;

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Сегментация по загрузке отелей

Анализ того, насколько эффективно используются разные типы номеров в разных отелях. Необходимо сопоставить фактически проданные ночи с максимально возможной емкостью отеля за определенный период.

Это сегментирование типов номеров по уровню их эффективности: от перегруженных до недоиспользуемых.

DECLARE @DaysInMonth INT = DAY(EOMONTH(GETDATE()));

WITH HotelCapacity AS (

    --сколько ночей может продать отель за @DaysInMonth дней

    SELECT

        h.hotel\_id,

        h.hotel\_name,

        rt.room\_type\_id,

        rt.type\_name,

        COUNT(r.room\_id) AS total\_rooms,

        --емкость за @DaysInMonth

        COUNT(r.room\_id) \* @DaysInMonth AS potential\_nights

    FROM prj\_hotels h

        JOIN prj\_rooms r ON h.hotel\_id = r.hotel\_id

        JOIN prj\_room\_types rt ON r.room\_type\_id = rt.room\_type\_id

    GROUP BY h.hotel\_id, h.hotel\_name, rt.room\_type\_id, rt.type\_name

),

ActualOccupancy AS (

    --сколько ночей было фактически прожито (из таблицы check\_info)

    SELECT

        r.hotel\_id,

        room.room\_type\_id,

        SUM(DATEDIFF(day, ci.check\_in\_fact, ISNULL(ci.check\_out\_fact, GETDATE()))) AS actual\_nights,

        AVG(p.price) AS avg\_real\_price

    FROM prj\_reservations r

        JOIN prj\_rooms room ON room.room\_id = r.room\_id

        JOIN prj\_check\_info ci ON r.reservation\_id = ci.reservation\_id

        JOIN prj\_hotel\_room\_prices p ON r.hotel\_id = p.hotel\_id AND room.room\_type\_id = p.room\_type\_id

    --анализ за последние @DaysInMonth дней

    WHERE ci.check\_in\_fact >= DATEADD(day, -@DaysInMonth, GETDATE())

    GROUP BY r.hotel\_id, room.room\_type\_id

)

SELECT

    hc.hotel\_name AS "Отель",

    hc.type\_name AS "Категория номера",

    hc.total\_rooms AS "Кол-во номеров",

    ISNULL(ao.actual\_nights, 0) AS "Продано ночей",

    hc.potential\_nights AS "Емкость за мес.",

    CAST(ISNULL(ao.actual\_nights, 0) \* 100.0 / hc.potential\_nights AS DECIMAL(10,2)) AS "Загрузка %",

    CASE

        WHEN (ISNULL(ao.actual\_nights, 0) \* 100.0 / hc.potential\_nights) > 35 THEN 'Высокий спрос (Пора повышать цены)'

        WHEN (ISNULL(ao.actual\_nights, 0) \* 100.0 / hc.potential\_nights) BETWEEN 30 AND 35 THEN 'Оптимально'

        WHEN (ISNULL(ao.actual\_nights, 0) \* 100.0 / hc.potential\_nights) BETWEEN 10 AND 29 THEN 'Недозагрузка (Нужны акции)'

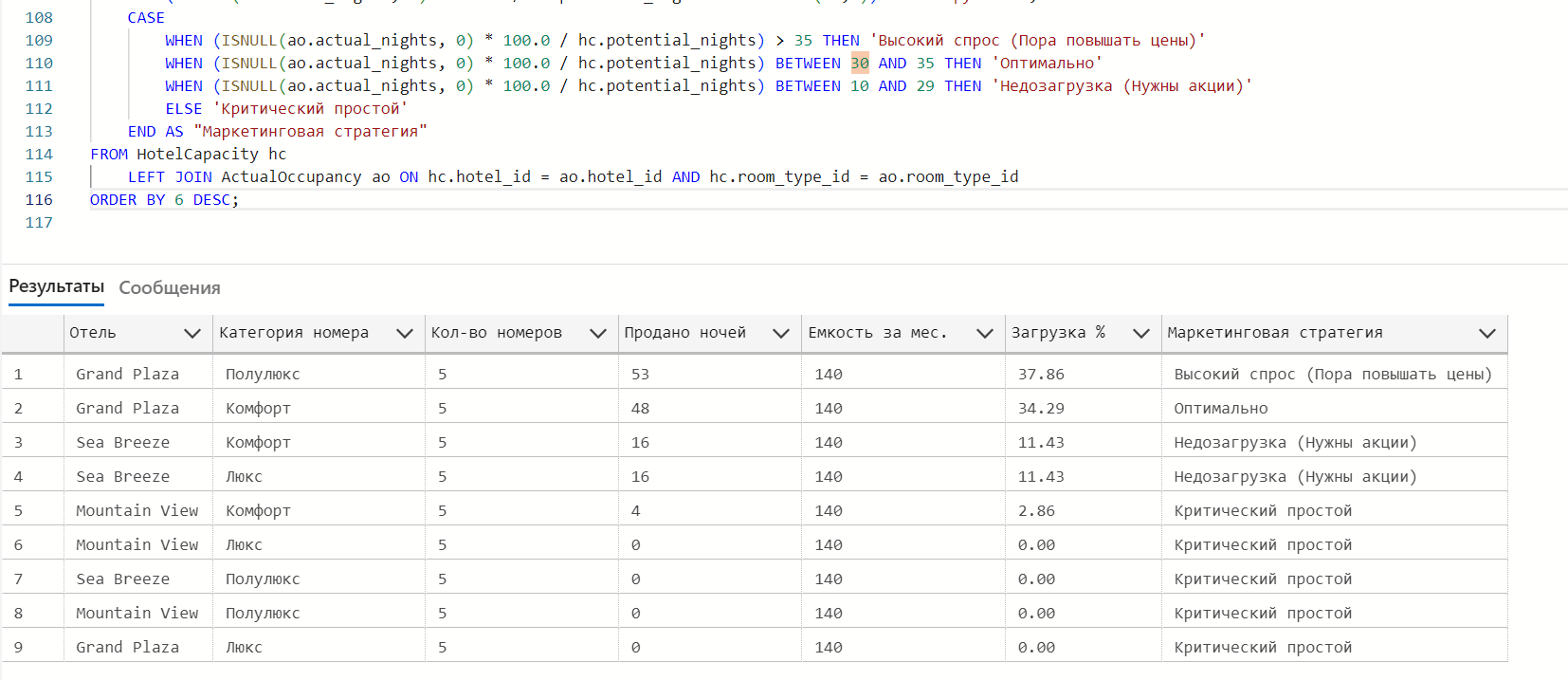
        ELSE 'Критический простой'

    END AS "Маркетинговая стратегия"

FROM HotelCapacity hc

    LEFT JOIN ActualOccupancy ao ON hc.hotel\_id = ao.hotel\_id AND hc.room\_type\_id = ao.room\_type\_id

ORDER BY 6 DESC;



Список всех забронированных номеров на сегодня с именами гостей