

Федеральное государственное автономное образовательное
учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Высшая школа бизнеса

КОНТРОЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Проектирование и разработка базы данных для продажи авиабилетов

по направлению подготовки 38.03.05
образовательная программа «Бизнес-
информатика»

Проект выполнили:

Патаев Арслан Зольванович

Москва 2024

Оглавление

Введение	3
Общие сведения о предметной области	3
Цель создания базы данных	3
Возможные пользователи базы данных	3
Возможные сценарии взаимодействия пользователей и БД	4
Бизнес-процесс	4
Список основных сущностей, атрибуты с указанием РК	6
Типы и описание связей между сущностями	13
Алгоритмы обработки данных, используемые в сценариях	18
Бизнес-функция	18
Планируемый перечень отчетных форм	18
Возможная архитектура программного продукта на основе нашей базы данных	18
Инфологическая модель	19
Даталогическая модель	19
Типы данных в физической модели	20
Соответствие таблиц 3НФ	21
Генерация и заполнение БД данными	22
Настройка индексов	28
Разработка запросов	30
Разработка триггера	46
Разработка функций	46
Разработка представлений	48
Разработка процедур	49
Информационная панель в Excel	54
Отчеты, визуализация	57
Описание роли каждого участника проекта	60

Введение

В последнее время в области теории баз данных было проведено несколько крупных исследований. Полученные результаты смело можно считать наиболее важным достижением информатики за последние 30 лет. Современные базы данных стали основой информационных систем и значительно изменили алгоритм работы большинства организаций. Развитие технологии баз данных способствовало созданию удобных и быстрых систем, что сделало их популярными среди широкого круга пользователей. В наше время большинство компаний сильно заинтересованы в создании информационных баз данных, так как они не только структурируют необходимую информацию, но и обеспечивают её быструю обработку. Наша команда задумала создать базу данных для удобной группировки и хранения информации о продаже билетов авиакомпаниями.

Общие сведения о предметной области

Компания N, наш заказчик, обратилась к нам с задачей разработки базы данных для удобной группировки и хранения информации о продаже билетов авиакомпаниями. До недавних изменений в законодательстве и экономической ситуации в мире ежедневно самолеты совершали около 120 тысяч полетов и перевозили примерно 12 миллионов пассажиров. В условиях изменившихся правил и требований к перелетам людям по-прежнему необходимо совершать различные перелеты между городами и странами по разным причинам, несмотря на сокращение количества рейсов. Это делает создание базы данных для компании N актуальным, так как она сможет оптимизировать процессы многих авиакомпаний. Предметная область включает в себя не только учёт продажи авиабилетов различными авиакомпаниями, но и хранение, анализ этой информации для развития компании в этой области в целом. Для этого необходимо хранить и обрабатывать информацию об аэропортах, рейсах, билетах, команде и персонале самолета, операциях и пассажирах. Из-за этой потребности компания N инициировала создание базы данных, которая быстро предоставит доступ к необходимой информации и поможет в принятии управленческих решений.

Цель создания базы данных

Главной целью создания базы данных для компании N является хранение информации о расписании авиаперелетов, о компаниях, которые их осуществляют, о самолетах этих компаний, о команде, работающей на каждом рейсе, о пассажирах, совершающих перелеты и покупающих билеты, и, конечно, о совершенных покупателями операциях. Сбор этой информации с последующим хранением и анализом не только оптимизирует процесс продажи билетов, но и позволяет компании N более точно рассчитывать показатели эффективности, прогнозировать спрос, улучшать логистические и операционные процессы, а также значительно уменьшать количество возможных ошибок как в продаже авиабилетов, так и в процессах самой компании.

Возможные пользователи базы данных

Возможными пользователями нашей базы данных могут стать любые компании, организации, их сайты или, например, сайты турагентств, ориентированные на продажу авиабилетов. В этот список входят и сами авиакомпании, нуждающиеся в анализе, отслеживании и хранении информации о проданных билетах. Благодаря нашей базе данных у них появится возможность иметь более структурированное и детальное представление о всех операциях, связанных с продажей авиабилетов, а также использовать данные для совершенствования внутренних процессов и расчета различных показателей эффективности.

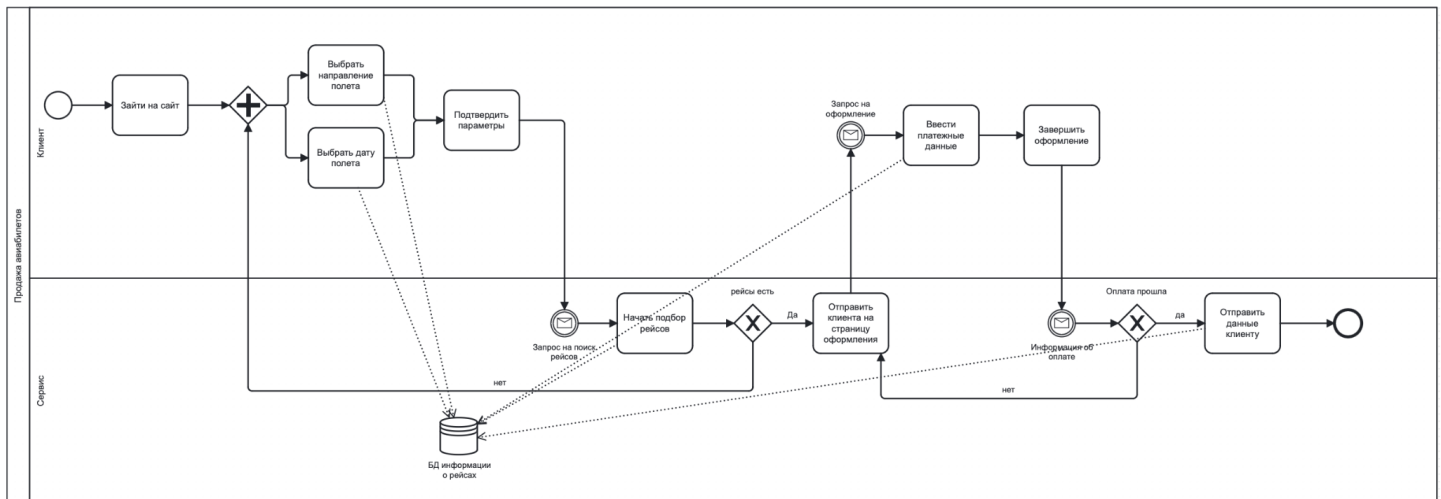
Возможные сценарии взаимодействия пользователей и БД

Одним из возможных сценариев взаимодействия может быть использование пользователем любого сайта по поиску авиабилетов. В данном процессе он выбирает конкретное направление полета, после чего ему предлагается выбрать дату и время вылета и прилета. Пользователь выбирает необходимый вариант, определяется с местом на борту рейса, вводит свои персональные данные и производит покупку. После этого происходит процесс обращения к базе данных и её работы. В неё заносится вся необходимая информация о пользователе, и происходит его добавление. Параллельно с этим выполняется процесс, который зависит от самого рейса, а именно, определяется самолет, состав персонала, экипажа и т. д. Дополнительно база данных может использоваться для анализа текущих и прошлых данных, что помогает компании N в расчете показателей эффективности, оптимизации маршрутов, управления запасами и планировании новых рейсов, а также в совершенствовании маркетинговых и логистических стратегий.

Бизнес-процесс

Один из примеров решения бизнес-задач с помощью внедрения базы данных в процесс продажи авиабилетов. Этот пример демонстрирует, как база данных может значительно улучшить управление бизнес-процессами и принятие управленческих решений в авиакомпании.

Процесс 1 – продажа авиабилетов BPMN :



Процесс начинается с того, что клиент заходит на сайт авиакомпании или агрегатора авиабилетов. На первой стадии клиент выбирает направление полета и дату поездки. В этот момент сайт отправляет запрос в базу данных для получения информации о доступных рейсах и свободных местах. База данных возвращает соответствующую информацию, и клиенту отображаются доступные рейсы, включая время вылета, прилета, стоимость билетов и доступные места. После этого клиент выбирает конкретный рейс и место в самолете. Выбранный рейс и место резервируются путем отправки запроса в базу данных. Затем клиент вводит свои персональные данные, которые также сохраняются в базе данных. Следующий этап — это оплата билета. Клиент переходит к оплате и, после успешного завершения транзакции, информация о ней сохраняется в базе данных. После этого база данных обновляет информацию о занятых местах, и клиент получает подтверждение бронирования, а также электронный билет.

Теперь рассмотрим, как база данных помогает в принятии управленческих решений. Во-первых, база данных собирает информацию о выбранных направлениях, датах и времени полетов. Это позволяет анализировать спрос на конкретные рейсы и оптимизировать расписание, что в свою очередь способствует более эффективному использованию ресурсов. Кроме того, с помощью данных о занятых местах можно управлять ценовой политикой, предлагая скидки на менее популярные рейсы или места. Это помогает повысить загрузку самолетов и, соответственно, увеличить доходы компании.

Информация о рейсах и пассажирах, собранная в базе данных, также помогает эффективно распределять ресурсы, такие как персонал на рейсах, обслуживание самолетов и наземные службы. Это позволяет улучшить операционную эффективность и качество обслуживания клиентов. Сохраненные данные о клиентах позволяют персонализировать предложения и улучшать качество обслуживания, что может повысить лояльность клиентов и привлечь новых. Анализ исторических данных дает возможность делать точные прогнозы и планировать будущие рейсы, что способствует стратегическому развитию компании. Наконец, сохраненные данные о транзакциях помогают контролировать финансовые потоки и принимать решения о финансовых стратегиях, что улучшает управление финансовыми ресурсами компании.

Внедрение базы данных не только автоматизирует процесс продажи билетов, но и предоставляет богатую информацию для принятия обоснованных управленческих решений. Это способствует улучшению операционной эффективности и конкурентоспособности компании, делая её более гибкой и готовой к изменениям на рынке.

Список основных сущностей, атрибуты с указанием PK

Таблица 1 – информация об аэропорте (описание атрибутов сущности AIRPORT)

:

Имя атрибута	Тип данных	Комментарий	Значение
id_airport	int	PK	ID аэропорта
name_airport	varchar(50)		Название аэропорта
duty_free	varchar(3)		Наличие duty free в аэропорту

Таблица 2 – тип аэропорта (описание атрибутов сущности TYPE_AIRPORT)

Имя атрибута	Тип данных	Комментарий	Значение
id_type_airport	int	PK	ID типа аэропорта
type_airport	varchar(40)		Тип аэропорта

Таблица 3 – информация о стране (описание атрибутов сущности COUNTRY):

Имя атрибута	Тип данных	Комментарий	Значение
id_country	int	PK	ID страны
name_country	varchar(40)		Название страны

Таблица 4 – информация о городе (описание атрибутов сущности TOWN):

Имя атрибута	Тип данных	Комментарий	Значение
id_town	int	PK	ID города
name_town	varchar(50)		Название города

Таблица 5 – информация о самолете (описание атрибутов сущности AIRPLANE):

Имя атрибута	Тип данных	Комментарий	Значение
id_plane	int	PK	ID самолета
name_plane	varchar(100)		Название самолёта

seats_plane	varchar(8)		Количество посадочных мест
-------------	------------	--	----------------------------

Таблица 6 – информация о модели самолета (описание атрибутов сущности MODEL_PLANE):

Имя атрибута	Тип данных	Комментарий	Значение
id_model_plane	int	PK	ID модели самолёта
model_plane	varchar(150)		Название модели самолёта

Таблица 7 – информация об экипаже (описание атрибутов сущности TEAM):

Имя атрибута	Тип данных	Комментарий	Значение
id_team	int	PK	ID команды
num_team	varchar(8)		Номер экипажа

Таблица 8 – информация о сотрудниках (описание атрибутов сущности EMPLOYEE):

Имя атрибута	Тип данных	Комментарий	Значение
id_emp	int	PK	ID сотрудника
l_name	varchar(30)		Фамилия сотрудника
f_name	varchar(30)		Имя сотрудника
birthday_emp	date		Дата рождения сотрудника

tel_emp	nvarchar(100)		Номер телефона сотрудника
---------	---------------	--	---------------------------

position	varchar(100)		Должность сотрудника
----------	--------------	--	----------------------

Таблица 9 – информация об авиакомпании (описание атрибутов сущности AIRLINE):

Имя атрибута	Тип данных	Комментарий	Значение
id_airline	int	PK	ID авиакомпании
airline	varchar(100)		Название авиакомпании
address_airline	varchar(300)		Адрес головного офиса авиакомпании

Таблица 10 – информация о рейсах (описание атрибутов сущности FLIGHT):

Имя атрибута	Тип данных	Комментарий	Значение
id_flight	int	PK	ID рейса
departure_id_airport	int		ID аэропорта вылета
arrival_id_airport	int		ID аэропорта прибытия
departure_date	datetime		Дата вылета
num_flight	varchar(9)		Номер рейса
arrival_date	datetime		Дата прилета
status_flight	varchar(100)		Статус рейса

Таблица 11 – информация о билетах (описание атрибутов сущности TICKET):

Имя атрибута	Тип данных	Комментарий	Значение
id_ticket	int	PK	ID билета
cost_ticket	int		Стоимость билета
is_sold	varchar(5)		Продажа билета (продан/ не продан)
num_ticket	int		Номер билета

Таблица 12 – информация о пассажирах (описание атрибутов сущности PASSENGER):

Имя атрибута	Тип данных	Комментарий	Значение
id_pas	int	PK	ID пассажира
last_name	varchar(100)		Фамилия пассажира
first_name	varchar(100)		Имя пассажира
email_pass	varchar(100)		email пассажира
birthday_pass	date		Дата рождения
sex	varchar(6)		Пол/статус пассажира

Таблица 13 – информация об операции, проведенной с билетом (описание атрибутов сущности OPERATION):

Имя атрибута	Тип данных	Комментарий	Значение
id_operation	int	PK	ID покупки
date_operation	date		Дата оплаты

Таблица 14 – информация о классе места (описание атрибутов сущности CLASS_PLACE):

Имя атрибута	Тип данных	Комментарий	Значение
id_class	int	PK	ID класса
name_class	varchar(50)		Название класса

Таблица 15 – Информация о месте в самолете (описание атрибутов сущности PLACES):

Имя атрибута	Тип данных	Комментарий	Значение
id_place	int	PK	ID места
num_place	varchar(8)		Номер места
num_row	varchar(8)		Номер ряда

Таблица 16 – информация о выполненной операции (описание атрибутов сущности TYPE_OPERATION):

Имя атрибута	Тип данных	Комментарий	Значение
id_type_operation	int	PK	ID операции

name_operation	varchar(100)		Название операции
----------------	--------------	--	-------------------

Таблица 17 – информация о распределении самолетов на каждый рейс
(описание атрибутов сущности flight_air) :

Имя атрибута	Тип данных	Комментарий	Значение
id_flight	int	PK	ID рейса
id_plane	int	PK	ID самолета

Таблица 18 – информация о распределении сотрудников по командам (описание атрибутов сущности team_emp) :

Имя атрибута	Тип данных	Комментарий	Значение
id_team	int	PK	ID команды
id_emp	int	PK	ID сотрудника

Типы и описание связей между сущностями

Таблица 19 – информация о типах связей между сущностями:

Название связи	Главная таблица	Дочерняя таблица	Тип связи	Идентифицирующая
flight_air	FLIGHT	AIRPLANE	(M:N)	см. ниже
team_emp	TEAM	EMPLOYEE	(M:N)	см.ниже

which model	MODEL_PLANE	AIRPLANE	(1:N)	нет
-------------	-------------	----------	-------	-----

seat in plane	MODEL_PLANE	PLACES	(1:N)	нет
fly	AIRPORT	FLIGHT	(1:N)	нет
locate	TOWN	AIRPORT	(1:N)	нет
belong to airline	AIRLINE	AIRPLANE	(1:N)	нет
domestic	COUNTRY	TOWN	(1:N)	нет
ticket operation	TICKET	OPERATION	(1:N)	нет
operation type	TYPE_OPERATION	OPERATION	(1:N)	нет
place class	CLASS_PLACE	PLACES	(1:N)	нет
ticket for the flight	FLIGHT	TICKET	(1:N)	нет
pas operation	PASSENGER	OPERATION	(1:N)	нет
has a type	TYPE_AIRPORT	AIRPORT	(1:N)	нет
team flight	FLIGHT	TEAM	(1:N)	нет

1. Связь "самолет в рейсе" между сущностями FLIGHT и AIRPLANE - (M:N), так как каждый самолет может участвовать в нескольких рейсах, а в каждом рейсе могут участвовать несколько самолетов. Участвующие в связи сущности равноправны, и такая связь не может идентифицировать экземпляры этих сущностей. Связь обязательна с обеих сторон, так как рейс невозможен без самолета, а самолет без рейса. Поэтому определение характера связи (M:N) не имеет смысла.

2. Связь "входит в команду" между сущностями TEAM и EMPLOYEE - (M:N),

поскольку каждый сотрудник может быть частью нескольких команд, а каждая команда состоит из нескольких сотрудников. Связь равноправная и не позволяет идентифицировать экземпляры сущностей. Связь обязательна с обеих сторон, так как команда невозможна без сотрудников, а сотрудники не могут существовать без команды. Определение характера связи (M:N) не имеет смысла.

3. Связь "имеет модель" между сущностями MODEL_PLANE и AIRPLANE - (1:N), так как каждый самолет имеет определенную модель, а моделей самолета может быть несколько. Связь не идентифицирующая, так как сущность AIRPLANE имеет первичный ключ, достаточный для идентификации. Идентифицирующая связь может привести к избыточности ключа. Участие сущности MODEL_PLANE в связи обязательно, а сущности AIRPLANE - нет.

4. Связь "места в самолете" между сущностями MODEL_PLANE и PLACES - (1:N), так как определенный набор мест закреплен за моделью самолета, но может принадлежать нескольким моделям. Связь не идентифицирующая (аналогично пункту 3). Участие сущности MODEL_PLANE в связи обязательно, а сущности PLACES - нет.

5. Связь "вылетают/прилетают" между сущностями AIRPORT и FLIGHT - (1:N), так как из одного аэропорта могут вылетать несколько рейсов, но каждый рейс вылетает только из одного аэропорта. Связь не идентифицирующая (аналогично пункту 3). Участие сущности AIRPORT в связи обязательно (рейс невозможен без аэропорта), а сущности FLIGHT - нет (аэропорт может существовать без рейсов).

6. Связь "находится в городе" между сущностями TOWN и AIRPORT - (1:N), так как каждый аэропорт находится в определенном городе, но в каждом городе может быть несколько аэропортов. Связь не идентифицирующая (аналогично пункту 3). Участие сущности TOWN в связи обязательно, а сущности AIRPORT - нет.

7. Связь "принадлежит авиакомпании" между сущностями AIRLINE и AIRPLANE - (1:N), так как каждый самолет принадлежит определенной авиакомпании, а у одной авиакомпании может быть несколько самолетов. Связь не идентифицирующая (аналогично пункту 3). Участие сущности AIRLINE в связи обязательно, а сущности AIRPLANE - нет.

8. Связь "находится в стране" между сущностями COUNTRY и TOWN - (1:N), так как каждый город находится в определенной стране, но в каждой стране может быть несколько городов. Связь не идентифицирующая (аналогично пункту 3). Участие сущности COUNTRY в связи обязательно, а сущности TOWN - нет.

9. Связь "операция с билетом" между сущностями TICKET и OPERATION - (1:N), так как каждой операции соответствует один билет, а по одному билету может пройти несколько операций. Связь не идентифицирующая (аналогично пункту 3). Участие сущности TICKET в связи обязательно, а сущности OPERATION - нет.

10. Связь "тип операции" между сущностями TYPE_OPERATION и OPERATION - (1:N), так как каждой операции соответствует один тип, но одному типу может принадлежать несколько операций. Связь не идентифицирующая (аналогично пункту 3). Участие сущности TYPE_OPERATION в связи обязательно, а сущности OPERATION - нет.

11. Связь между сущностями CLASS_PLACE и PLACES - (1:N), так как у каждого места есть определенный класс, а каждому классу принадлежит несколько мест. Связь не идентифицирующая (аналогично пункту 3). Участие сущности CLASS_PLACE в связи обязательно, а сущности PLACES - нет.

12. Связь "билет на полет" между сущностями FLIGHT и TICKET - (1:N), так как каждому билету соответствует один полет, но на один полет может быть продано много билетов. Связь не идентифицирующая (аналогично пункту 3). Участие сущности FLIGHT в связи обязательно, а сущности TICKET - нет.

13. Связь "операции пассажиров" между сущностями PASSANGER и OPERATION - (1:N), так как каждой операции соответствует один пассажир, а один пассажир может выполнить несколько операций. Связь не идентифицирующая (аналогично пункту 3). Участие сущности PASSANGER в связи обязательно, а сущности OPERATION - нет.

14. Связь "аэропорт имеет тип" между сущностями TYPE_AIRPORT и AIRPORT - (1:N), так как каждый аэропорт имеет тип, но каждый тип может принадлежать нескольким аэропортам. Связь не идентифицирующая (аналогично пункту 3). Участие сущности TYPE_AIRPORT в связи обязательно, а сущности AIRPORT - нет.

15. Связь "команда полета" между сущностями FLIGHT и TEAM - (1:N), так как в каждом перелете участвует определенная команда, но одна команда может участвовать в нескольких перелетах. Связь не идентифицирующая (аналогично пункту 3). Участие сущности FLIGHT в связи обязательно, а сущности TEAM - нет.

Алгоритмы обработки данных, используемые в сценариях

В процессе выполнения данного проекта для компании N мы используем стандартные для большинства СУБД SQL-запросы в качестве основного инструмента обработки данных. Кроме того, мы задействуем хранимые процедуры, индексы, функции и триггеры, чтобы обеспечить более качественную и эффективную обработку данных.

Бизнес-функция

Главная бизнес-функция заключается в учете продаж авиабилетов и получении различной статистики, охватывающей все этапы процесса покупки авиабилетов для различных авиакомпаний.

Планируемый перечень отчетных форм

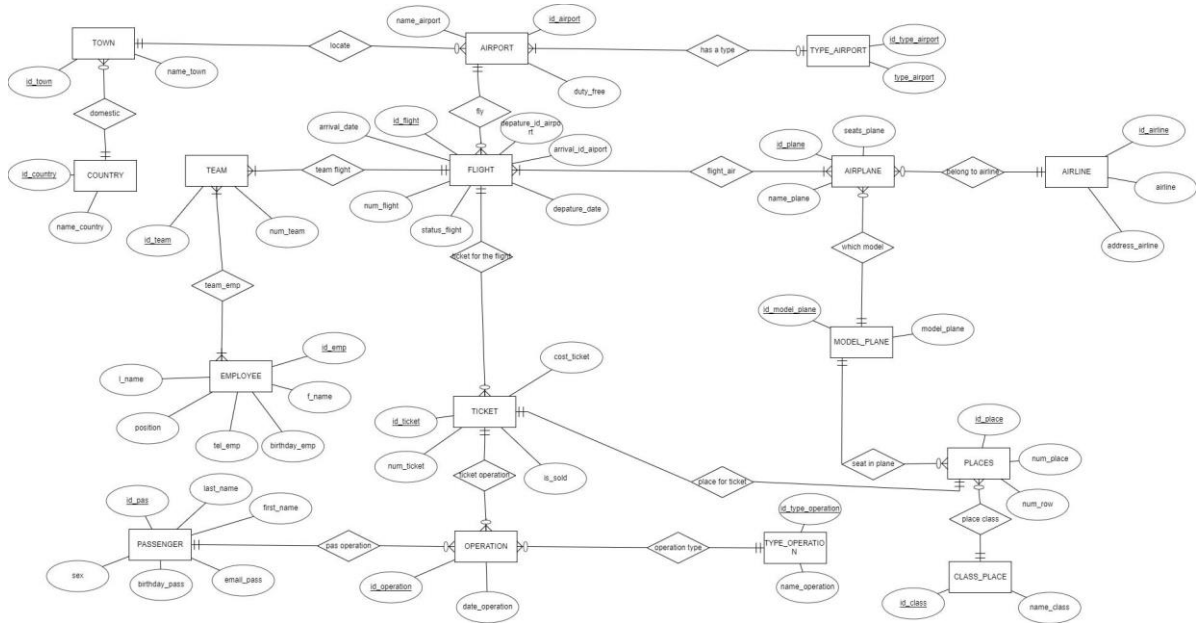
В результате нашей работы компания N сможет получать разнообразную статистику по различным параметрам. Например, можно будет узнать, какой город является наиболее популярным для перелетов среди клиентов авиакомпаний, какова средняя цена билета, сколько в среднем людей покупают авиабилеты в конкретный день, по какой цене чаще всего приобретаются билеты, а также какой класс перелета имеет больший спрос среди клиентов. На основе этой информации можно будет корректировать цены на определенные направления перелетов, а также в определенные дни месяца. Статистика будет извлекаться и предоставляться пользователям нашей базы данных в конкретной форме, например, в виде диаграммы, графика или таблицы, которые будут созданы с помощью инструмента визуализации данных.

Возможная архитектура программного продукта на основе нашей базы данных

Архитектура программного продукта на основе нашей базы данных будет ориентирована на клиентов авиакомпаний, которые обращаются к организации с целью приобретения авиабилета. Потребитель будет выбирать город назначения, класс перелета, место в самолете, а также вводить свои персональные данные, необходимые для покупки авиабилета. Далее, приложение будет использовать разработанную нами базу данных для предоставления всех возможных вариантов выбора и передавать эту информацию клиенту.

Инфологическая модель

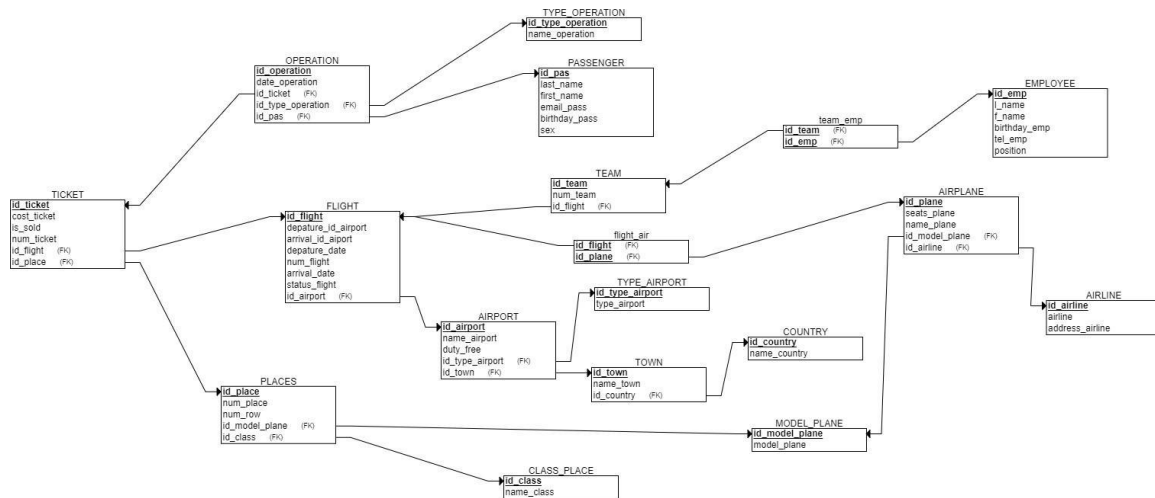
С помощью инструмента ERDPlus мы разработали инфологическую модель, которая дает возможность наглядно увидеть описание структуры всей задействованной области:



Инфологическая модель

Даталогическая модель

В контексте СУБД данная модель представляет собой инфологическую, показывает данные и связи между ними:

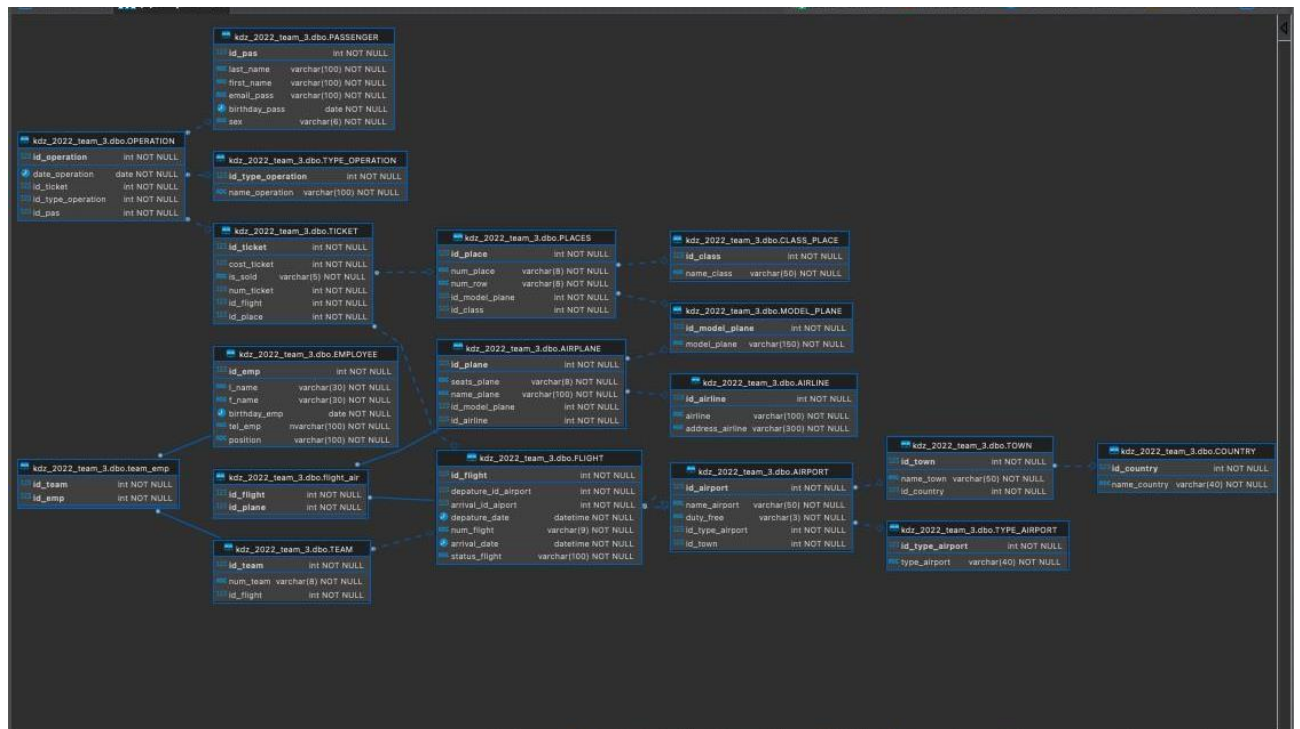


Даталогическая модель

В таблице с названием FLIGHT для нашей БД, следующие FK: id_departure_airport (int), id_arrival_airport (int). В таблицах team_emp и flight_air составной первичный ключ.

Типы данных в физической модели

Типы данных показаны с помощью диаграммы:



Первоначальная диаграмма

Диаграмма ниже представляет собой конечную версию, после проведенной работы (а именно, написания процедур, запросов, и т.д.):

PASSENGER	-	+	+	-	+
AIRPORT	-	+	+	-	+
FLIGHT	-	+	+	-	+
OPERATION	-	+	+	-	+
TYPE_AIRPORT	-	+	+	-	+
AIRPLANE	-	+	+	-	+
MODEL_PLANE	-	+	+	-	+
TYPE_OPERATION	-	+	+	-	+
CLASS_PLACE	-	+	+	-	+
PLACES	-	+	+	-	+
AIRLINE	-	+	+	-	+
TICKET	-	+	+	-	+
FLIGHT_AIR	-	+	+	-	+
TEAM_EMP	-	+	+	-	+

Генерация и заполнение БД данными

В соответствии с критериями команда заполнила таблицы базы данными. Так, таблица PASSENGER содержит больше 1000 строк (1111). Каждая из таблиц заполнена случайно сгенерированными данными. Для этого мы использовали библиотеки Python. Примеры заполнения таблиц и фрагменты кода:

```

import pandas as pd
import random
import string
from faker import Faker

fake = Faker()

def generate_random_email():
    prefix = ''.join(random.choices(string.ascii_lowercase + string.digits, k=10))
    return f"{prefix}@example.com"

def generate_sex_title():
    titles = ['Mr.', 'Mrs.', 'Miss', 'Ms.', 'Dr.', 'Prof.']
    return random.choice(titles)

data = {
    'id_pas': [random.randint(1000, 2000) for _ in range(27)],
    'last_name': [fake.last_name() for _ in range(27)],
    'first_name': [fake.first_name() for _ in range(27)],
    'email_pass': [generate_random_email() for _ in range(27)],
    'birthday_pass': [fake.date_of_birth(minimum_age=0, maximum_age=100).strftime('%Y-%m-%d') for _ in range(27)],
    'sex': [generate_sex_title() for _ in range(27)]
}

df = pd.DataFrame(data)

df

```

	id_pas	last_name	first_name	email_pass	birthday_pass	sex
0	1714	Boyd	Kyle	xrklo5dogn@example.com	2021-08-23	Miss
1	1339	Hall	Stephanie	vzo684jx7c@example.com	1962-10-14	Ms.
2	1127	Ballard	David	xpp9gazhfk@example.com	1969-07-25	Dr.

Введите SQL выражение чтобы отфильтровать результаты							
Таблица	id_pas	last_name	first_name	email_pass	birthday_pass	sex	
1085	1 085	Stroman	Rigoberto	melvina.tillman@example.com	2019-02-09	Mrs.	
1086	1 086	Nolan	Dagmar	rlarson@example.net	1997-11-28	Prof.	
1087	1 087	Carter	Kaitlin	king.niko@example.com	1989-06-29	Mrs.	
1088	1 088	Cummerata	Alia	mrohan@example.net	1984-02-22	Mr.	
1089	1 089	Koss	Mose	asimonis@example.com	1994-09-21	Prof.	
1090	1 090	Dibbert	Rudy	justine05@example.org	1975-07-24	Dr.	
1091	1 091	Walsh	Jovanny	joseph34@example.com	1974-04-23	Dr.	
1092	1 092	Aufderhar	Arlo	yharber@example.net	2013-01-16	Miss	
1093	1 093	Schmidt	Lorenz	mziemmann@example.net	1989-08-12	Dr.	
1094	1 094	Zboncak	Elenora	osbaldo91@example.net	1980-02-05	Ms.	
1095	1 095	Tromp	Dax	lillie34@example.com	1987-03-07	Mr.	
1096	1 096	Moore	Jackie	ali.franecki@example.org	2000-08-08	Ms.	
1097	1 097	Gottlieb	Dillan	kcorkery@example.org	2016-12-06	Ms.	
1098	1 098	Schumm	Kaylin	amore@example.org	2008-04-03	Prof.	
1099	1 099	Frami	Alda	chelsey.johnston@example.net	2019-10-12	Dr.	
1100	1 100	Connelly	Micheal	kirk14@example.net	1998-08-28	Prof.	
1101	1 101	Swaniawski	Paris	wanda62@example.com	1994-08-07	Mr.	
1102	1 102	Gaylord	Koby	haag.dayton@example.net	2009-10-27	Prof.	
1103	1 103	Weber	Jerrold	deckow.barbara@example.org	2010-06-20	Dr.	
1104	1 104	Barrows	Bernita	nikko81@example.com	2008-07-22	Prof.	
1105	1 105	Lynch	Camylle	batz.fay@example.com	1977-10-05	Miss	
1106	1 106	Stracke	Albertha	francis34@example.org	1979-05-15	Mr.	
1107	1 107	Stamm	Anabel	cale61@example.net	2011-06-24	Mr.	
1108	1 108	Berge	Abbey	deborah45@example.org	1992-12-11	Miss	
1109	1 109	White	Felipa	ofelia.swaniawski@example.net	1973-12-19	Prof.	
1110	1 110	Yost	Karen	tbeahan@example.org	2002-02-19	Prof.	
1111	1 111	Hermiston	Annetta	carter.joanne@example.com	2007-09-19	Mr.	
Запись							
Save Cancel Script 200 1 111 Rows: 1 1111 строк получено - 49ms (+103ms)							

Пример заполнения таблицы PASSENGER

Ниже представлены примеры заполнения еще нескольких таблиц:

Введите SQL выражение чтобы отфильтровать результаты					
	id_airport	name_airport	duty_free	id_type_airport	id_town
1	1	Vienna International Airport	да	1	1
2	2	Sydney International Airport	да	2	2
3	3	Международный аэропорт Домодедово	да	3	3
4	4	Международный аэропорт Шереметьево	да	1	3
5	5	Международный аэропорт Внуково	да	2	3
6	6	Международный аэропорт Пулково	да	3	4
7	7	Международный аэропорт Сочи	да	1	5
8	8	Международный аэропорт Краснодар	да	2	6
9	9	Международный аэропорт Толмачево	да	3	7
10	10	Международный аэропорт Казань	да	1	8
11	11	Международный аэропорт Хабарово	да	2	9
12	12	Международный аэропорт Курумоч	да	3	10
13	13	Международный аэропорт Иркутск	нет	1	11
14	14	Международный аэропорт Киев	да	2	12
15	15	Международный аэропорт Днепропетровск	да	3	13
16	16	Международный аэропорт Донецк	нет	1	14
17	17	Национальный аэропорт Минск	нет	2	15
18	18	Toronto Pearson International Airport	да	3	16
19	19	John F. Kennedy International Airport	да	1	17
20	20	Washington Dulles International Airport	да	2	18
21	21	Miami International Airport	да	3	19


```

airport_names = [
    "Международный аэропорт Домодедово",
    "Международный аэропорт Шереметьево", "Международный аэропорт Внуково", "Международный аэропорт Краснодар", "Международный аэропорт Толмачево",
    "Международный аэропорт Казань", "Международный аэропорт Хабарово", "Международный аэропорт Курумоч", "Международный аэропорт Иркутск",
    "Международный аэропорт Киев", "Международный аэропорт Днепропетровск", "Международный аэропорт Донецк", "Национальный аэропорт Минск"
]

def generate_duty_free():
    return random.choice(["да", "нет"])

data = [
    {'id_airport': list(range(1, len(airport_names) + 1)),
     'name_airport': random.choices(airport_names, k=len(airport_names)),
     'duty_free': [generate_duty_free() for _ in range(len(airport_names))],
     'id_type_airport': [random.randint(1, 3) for _ in range(len(airport_names))],
     'id_town': [random.randint(1, 20) for _ in range(len(airport_names))]}
]

```


	id_airport	name_airport	duty_free	id_type_airport	id_town
0	1	Международный аэропорт Краснодар	нет	1	1
1	2	Международный аэропорт Пулково	нет	3	6

Пример заполнения таблицы AIRPORT

EMPLOYEE Введите SQL выражение чтобы отфильтровать результаты						
	id_emp	l_name	f_name	birthday_emp	tel_emp	position
1	1	Эванс	Оливия	1986-02-13	79743091736	Бортпроводник
2	2	Робертс	Эмма	1992-04-12	79005342878	Бортпроводник
3	3	Льюиз	София	1995-08-17	78940972645	Бортпроводник
4	4	Мартин	Оливер	1998-03-22	75940374651	Пилот
5	5	Уильямс	Томас	1968-06-19	73490615346	Пилот
6	6	Флоренс	Мия	1994-09-29	71034894682	Бортпроводник
7	7	Морган	Шарлотта	1999-05-30	70046284904	Бортпроводник
8	8	Стоун	Эмили	1999-07-28	73230957372	Бортпроводник
9	9	Грант	Карл	1974-09-15	73620467394	Пилот
10	10	Гибсон	Джон	1980-10-14	72309865429	Пилот
11	11	Елагина	Алина	1997-07-17	71238904675	Бортпроводник
12	12	Добрынина	Анна	2000-06-11	78290986453	Бортпроводник
13	13	Дмитриева	Олеся	2001-09-10	74039820312	Бортпроводник
14	14	Демидов	Иван	1988-10-10	78943023409	Пилот
15	15	Данилов	Анатолий	1975-11-18	70493876512	Пилот
16	16	Гурская	Екатерина	2000-11-13	74320965489	Бортпроводник
17	17	Волочкова	Ангелина	2000-12-12	74320957832	Бортпроводник
18	18	Владова	Анастасия	1998-08-16	79020645373	Бортпроводник
19	19	Ветров	Александр	1980-04-09	73109432094	Пилот
20	20	Бакурин	Кирилл	1981-03-03	74321230947	Пилот
21	21	Барина	Полина	2000-08-06	74120943287	Бортпроводник
22	22	Бабочкина	Мария	1999-09-05	74320958273	Бортпроводник
23	23	Игнатова	Елизавета	1998-09-03	74201856323	Бортпроводник
24	24	Иванов	Александр	1988-08-08	75309871234	Пилот
25	25	Сидоров	Алексей	1970-04-14	74320985234	Пилот

```
data = {
    'id_emp': list(range(1, 26)),
    'l_name': [fake.last_name() for _ in range(25)],
    'f_name': [fake.first_name() for _ in range(25)],
    'birthday_emp': [fake.date_of_birth(minimum_age=18, maximum_age=65).strftime('%Y-%m-%d') for _ in range(25)],
    'tel_emp': [fake.phone_number() for _ in range(25)],
    'position': [random.choice(positions) for _ in range(25)]
}

df = pd.DataFrame(data)
df
```

	id_emp	l_name	f_name	birthday_emp	tel_emp	position
0	1	Королева	Эммануил	1970-02-26	+70734528268	Бортпроводник
1	2	Овчинников	Януарий	2004-01-08	+73329219706	Пилот
2	3	Власова	Всеволод	1983-11-15	+7 (056) 831-37-35	Пилот

Пример заполнения таблицы EMPLOYEE

TICKET Введите SQL выражение чтобы отфильтровать результаты						
	123 id_ticket	123 cost_ticket	ABC is_sold	123 num_ticket	123 id_flight	123 id_place
1	1	2 400	да	1 000	1	1
2	2	3 000	да	1 001	2	2
3	3	12 000	да	1 002	3	8
4	4	25 000	нет	1 003	4	10
5	5	300	да	1 004	5	11
6	6	3 000	нет	1 005	6	12
7	7	15 000	нет	1 006	7	18
8	8	4 000	да	1 007	8	16
9	9	28 000	нет	1 008	9	20
10	10	2 400	да	1 009	10	17
11	11	3 500	нет	1 010	11	21
12	12	4 500	да	1 011	12	22
13	13	5 000	нет	1 012	13	23
14	14	30 000	да	1 013	14	31
15	15	35 000	нет	1 014	15	32
16	16	3 000	да	1 015	1	23
17	17	2 400	нет	1 016	2	25
18	18	2 400	да	1 017	3	24
19	19	2 400	да	1 018	4	26
20	20	10 000	нет	1 019	5	29
21	21	10 000	да	1 020	6	30
22	22	2 200	да	1 021	7	41

```
def generate_is_sold():
    return random.choice(["да", "нет"])

data = {
    'id_ticket': list(range(1, 26)),
    'cost_ticket': [random.randint(300, 35000) for _ in range(25)],
    'is_sold': [generate_is_sold() for _ in range(25)],
    'num_ticket': [1000 + i for i in range(25)],
    'id_flight': [random.randint(1, 20) for _ in range(25)],
    'id_place': [random.randint(1, 30) for _ in range(25)]
}
```

```
df = pd.DataFrame(data)
```

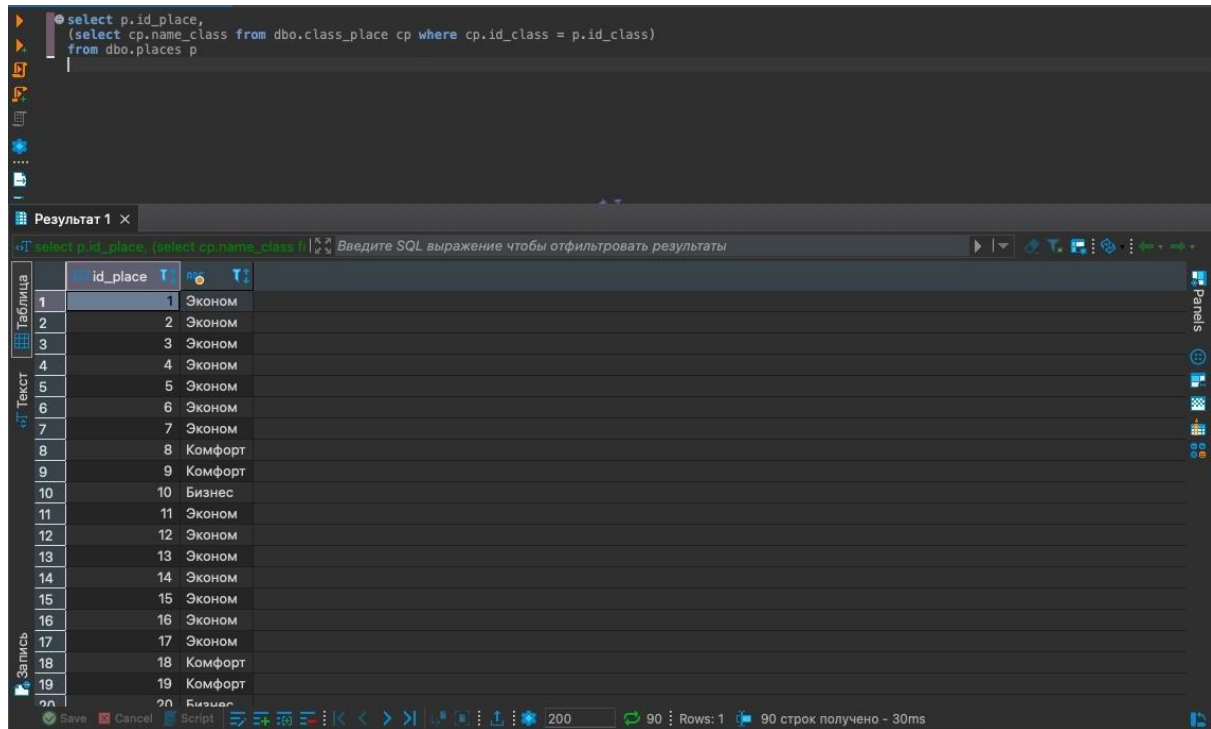
```
df
```

	id_ticket	cost_ticket	is_sold	num_ticket	id_flight	id_place
0	1	25989	нет	1000	2	15
1	2	11046	да	1001	8	9
2	3	27632	да	1002	14	3

Пример заполнения таблицы TICKET

Настройка индексов

Для настройки индекса используем запрос №3 (см. ниже). Индекс будет значительно сокращать время выполнения запроса, тем самым, оптимизируя процесс. Первоначальное время:



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

```
select p.id_place,
(select cp.name_class from dbo.class_place cp where cp.id_class = p.id_class)
from dbo.places p
```

The results pane shows a table with 20 rows. The first column is 'id_place' and the second column is 'name_class'.

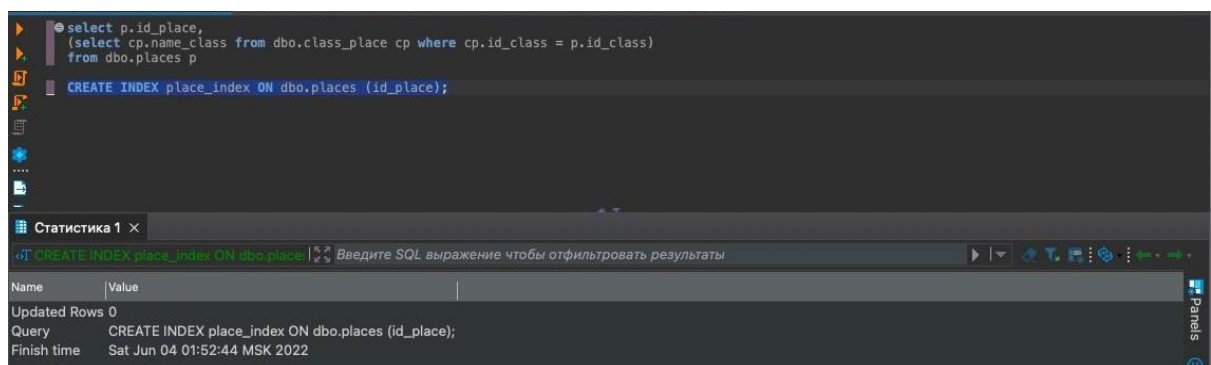
id_place	name_class
1	Эконом
2	Эконом
3	Эконом
4	Эконом
5	Эконом
6	Эконом
7	Эконом
8	Комфорт
9	Комфорт
10	Бизнес
11	Эконом
12	Эконом
13	Эконом
14	Эконом
15	Эконом
16	Эконом
17	Эконом
18	Комфорт
19	Комфорт
20	Бизнес

The status bar at the bottom indicates: 200 rows, 90 rows returned, 30ms execution time.

Первоначальное время обработки запроса

Создание индекса:

```
CREATE INDEX place_index ON dbo.places (id_place);
```



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

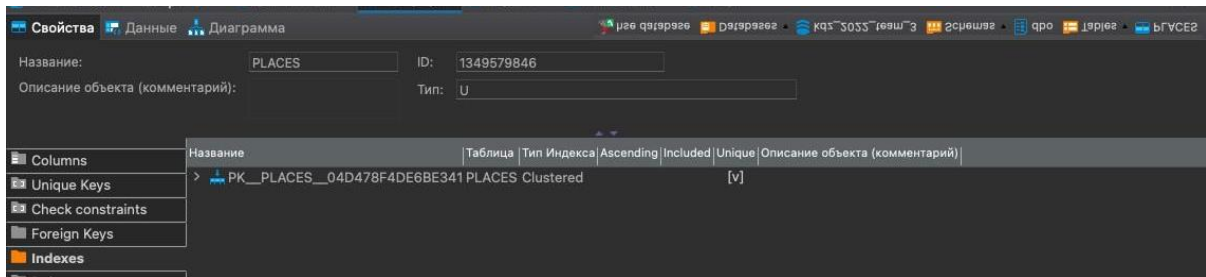
```
CREATE INDEX place_index ON dbo.places (id_place);
```

The results pane shows a table with 20 rows. The first column is 'id_place' and the second column is 'name_class'.

Name	Value
Updated Rows	0
Query	CREATE INDEX place_index ON dbo.places (id_place);
Finish time	Sat Jun 04 01:52:44 MSK 2022

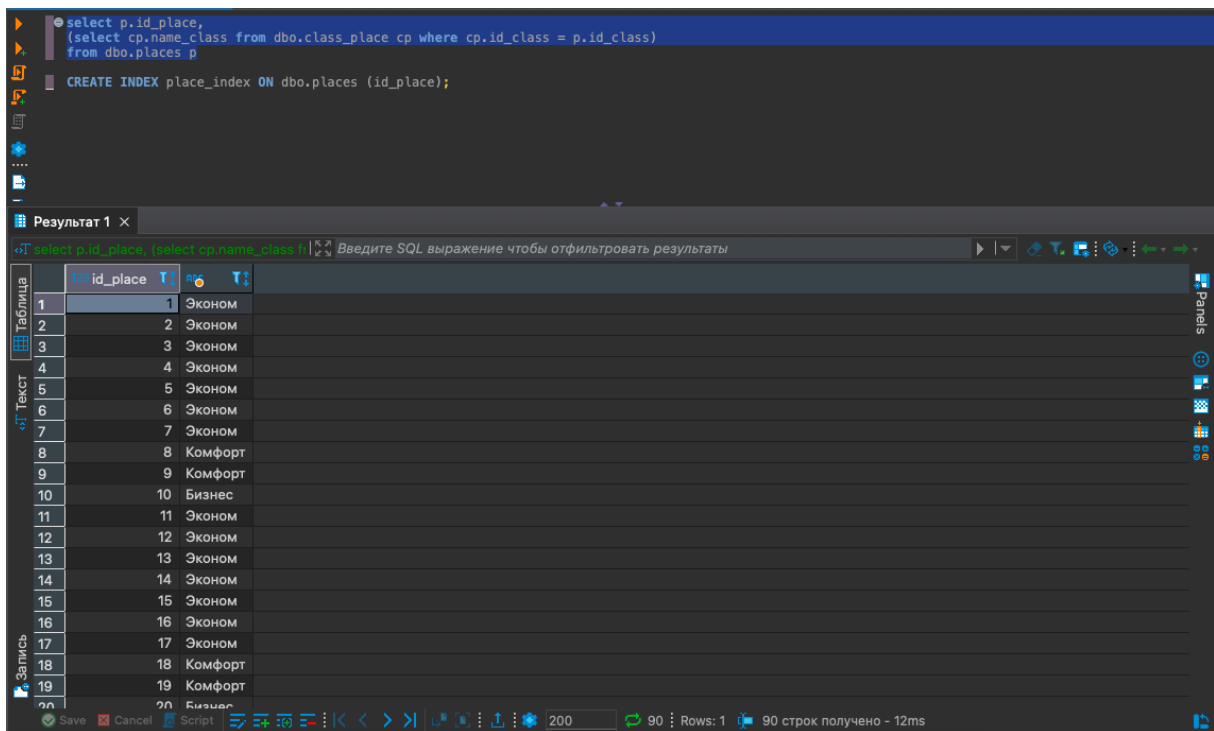
Создание индекса

Индекс:



Наличие индекса в таблице PLACES

Время обработки после создания индекса:

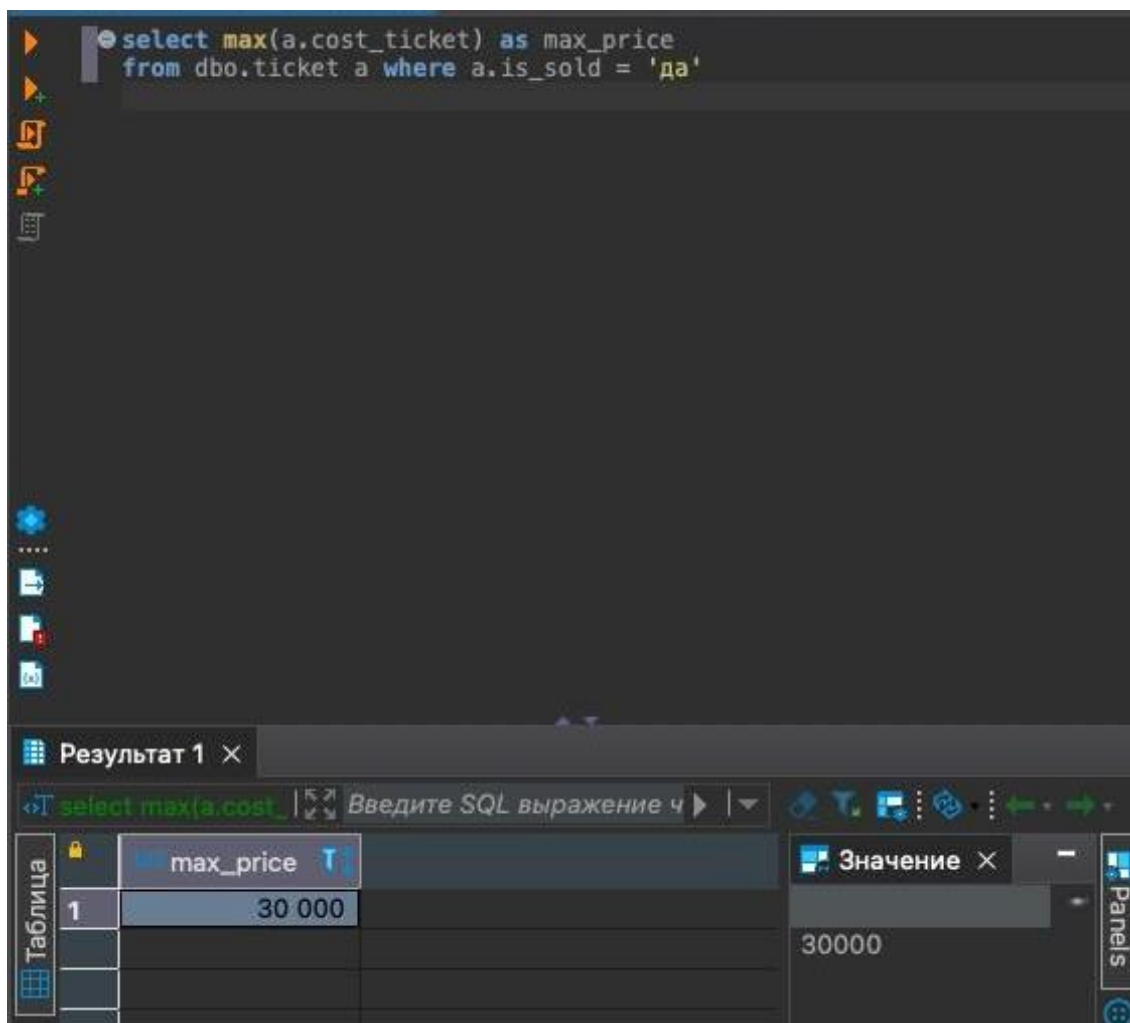


Конечное время обработки запроса

Разработка запросов

1) Вывод максимальной цены приобретенного билета (простой запрос с условием и формулами в SELECT):

```
select max(a.cost_ticket) as max_price  
from dbo.ticket a where a.is_sold = 'да'
```



Запрос №1

2) Вывод количества аэропортов, где нет duty free (простой запрос с условием и формулами в SELECT):

```
select count(a.id_airport)  
from dbo.airport a where a.duty_free = 'нет'
```

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

```
select count(a.id_airport)
from dbo.airport a where a.duty_free = 'нет'
```

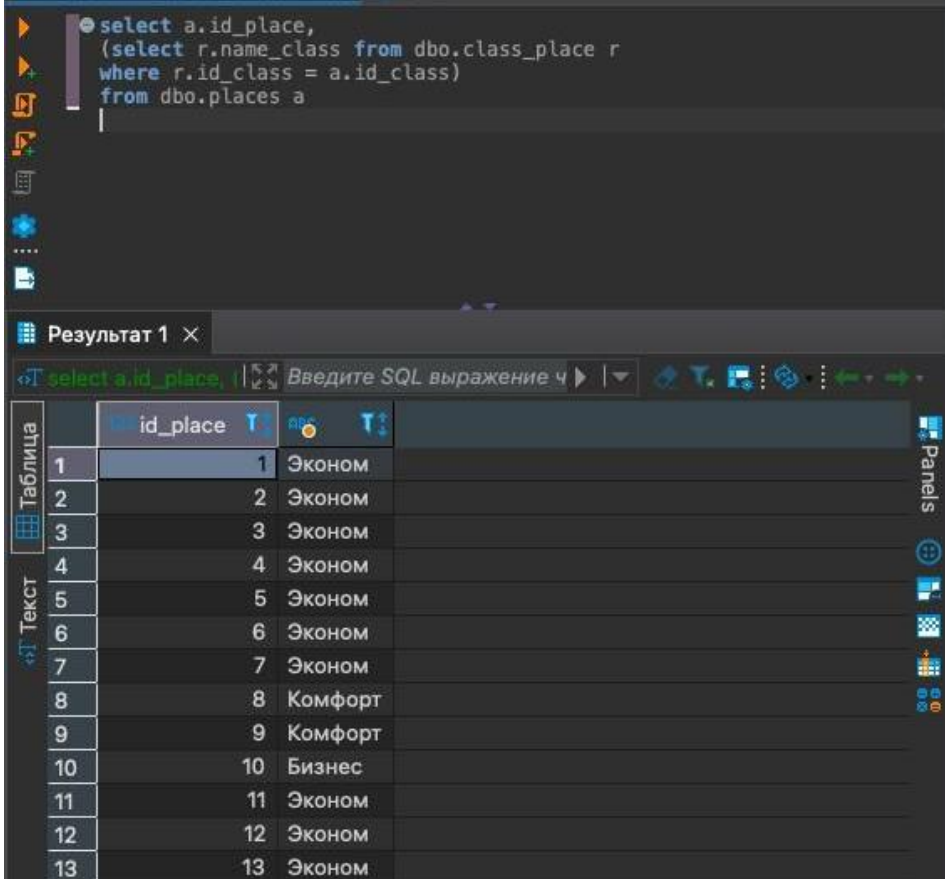
Below the query editor, the results are displayed in a table. The table has two columns: "Таблица" (Table) and "Значение" (Value). The first row shows the result of the query, which is 3.

Таблица	Значение
1	3

Запрос №2

3) Вывод класса для каждого места (запрос с коррелированным подзапросом в SELECT):

```
select a.id_place,  
(select r.name_class from dbo.class_place r where r.id_class =  
a.id_class) from dbo.places a
```



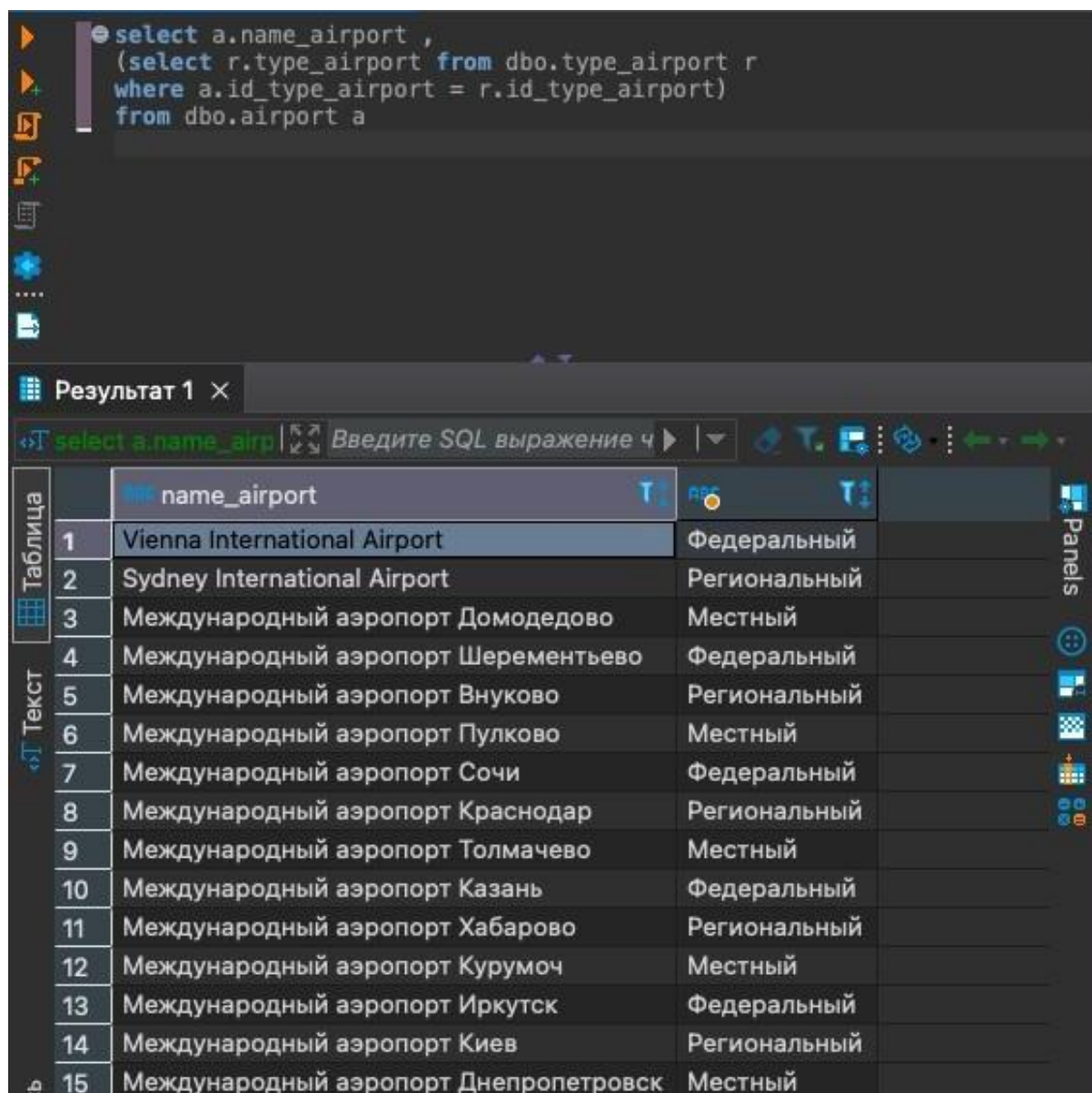
The screenshot shows a SQL query editor with a dark theme. The query is: `select a.id_place, (select r.name_class from dbo.class_place r where r.id_class = a.id_class) from dbo.places a`. Below the query, the results are displayed in a table grid. The table has two columns: `id_place` and `name_class`. The results are as follows:

id_place	name_class
1	Эконом
2	Эконом
3	Эконом
4	Эконом
5	Эконом
6	Эконом
7	Эконом
8	Комфорт
9	Комфорт
10	Бизнес
11	Эконом
12	Эконом
13	Эконом

Запрос №3

4) Вывод типа для каждого аэропорта (запрос с коррелированным подзапросом в SELECT):

```
select a.name_airport ,  
(select r.type_airport from dbo.type_airport r where a.id_type_airport =  
r.id_type_airport) from dbo.airport a
```

The screenshot shows a SQL query execution interface. At the top, a query is entered in a text area. Below it, the results are displayed in a table. The table has two columns: 'name_airport' and a column with Russian airport types. The results list 15 airports, alternating between international and domestic types.

Query:

```
select a.name_airport ,
(select r.type_airport from dbo.type_airport r
where a.id_type_airport = r.id_type_airport)
from dbo.airport a
```

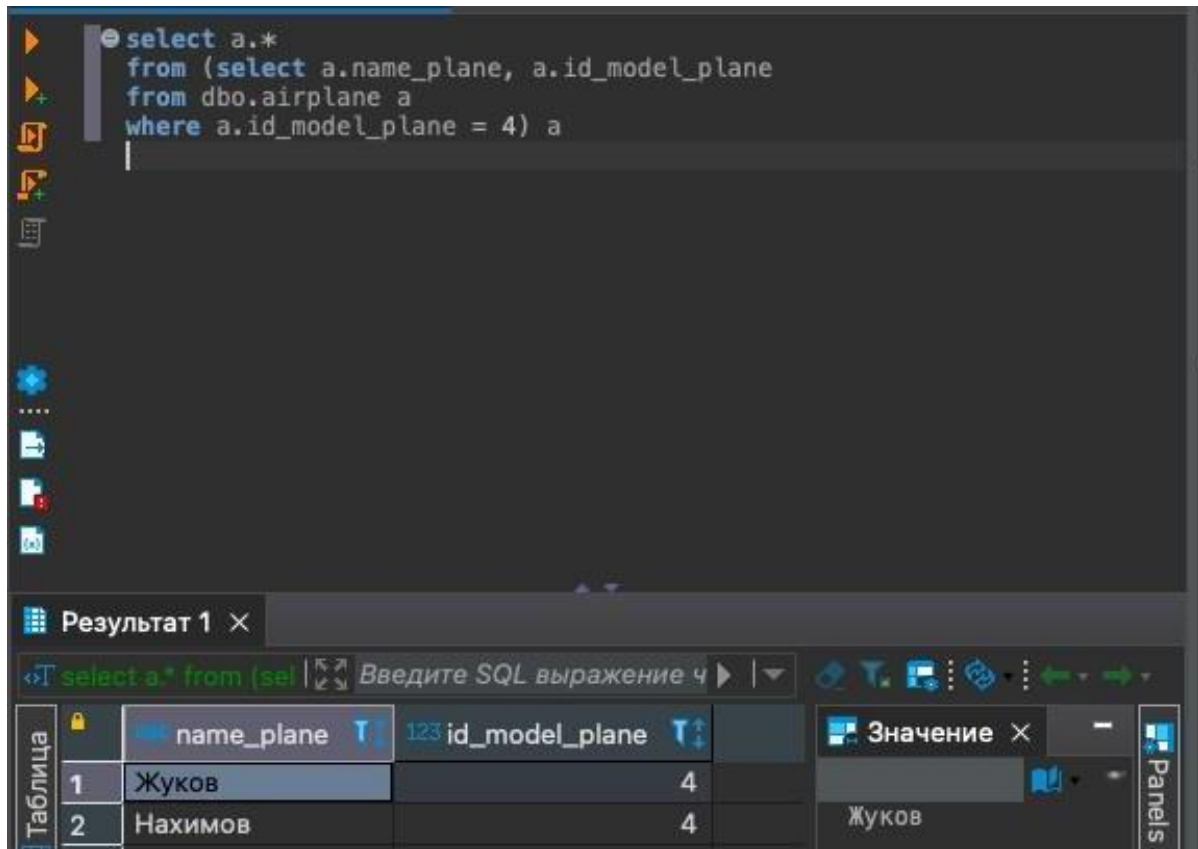
Result 1:

	name_airport	
1	Vienna International Airport	Федеральный
2	Sydney International Airport	Региональный
3	Международный аэропорт Домодедово	Местный
4	Международный аэропорт Шереметьево	Федеральный
5	Международный аэропорт Внуково	Региональный
6	Международный аэропорт Пулково	Местный
7	Международный аэропорт Сочи	Федеральный
8	Международный аэропорт Краснодар	Региональный
9	Международный аэропорт Толмачево	Местный
10	Международный аэропорт Казань	Федеральный
11	Международный аэропорт Хабарово	Региональный
12	Международный аэропорт Курумоч	Местный
13	Международный аэропорт Иркутск	Федеральный
14	Международный аэропорт Киев	Региональный
15	Международный аэропорт Днепропетровск	Местный

Запрос №4

5) Вывод именных названий самолетов модели Boeing 707 (запрос с подзапросом в FROM):

```
select a.*
from (select a.name_plane, a.id_model_plane from dbo.airplane a
where a.id_model_plane = 4) a
```



Запрос №5

6) Вывод имен, фамилий, дат рождений всех пилотов (запрос с подзапросом в FROM): select a.*

from (select e.l_name, e.f_name, e.birthdate from dbo.employee e
where e.[position] = 'Пилот') a

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a query in the query editor:

```
select a.*
from
(select e.l_name, e.f_name, e.birthday_emp
from dbo.employee e
where e.[position] = 'Пилот') a
```

The bottom pane shows the results of the query, labeled "Результат 1". The results are displayed in a table with the following columns: l_name, f_name, birthday_emp, and Значение. The table contains 9 rows of data:

	l_name	f_name	birthday_emp	Значение
1	Мартин	Оливер	1998-03-20	Мартин
2	Уильямс	Томас	1968-06-10	
3	Грант	Карл	1974-09-10	
4	Гибсон	Джон	1980-10-10	
5	Демидов	Иван	1988-10-10	
6	Данилов	Анатолий	1975-11-10	
7	Ветров	Александр	1980-04-00	
8	Бакурин	Кирилл	1981-03-00	
9	Иванов	Александр	1988-08-00	

Запрос №6

7) Вывод аэропорта и времени крайнего вылета из этого аэропорта для завершенных полетов (запрос с подзапросом в FROM, агрегированием, группировкой и сортировкой):

```
select a.*
from (select r.departure_id_airport, max(r.departure_date) as last_dep
from dbo.flight r where r.status_flight = 'Окончен' group by r.departure_id_airport)
a order by a.departure_id_airport
```

```
select a.*
from
(select r.depature_id_airport, max(r.depature_date) as last_dep
from dbo.flight r
where r.status_flight = 'Окончен'
group by r.depature_id_airport) a
order by a.depature_id_airport
```

Результат 1 X

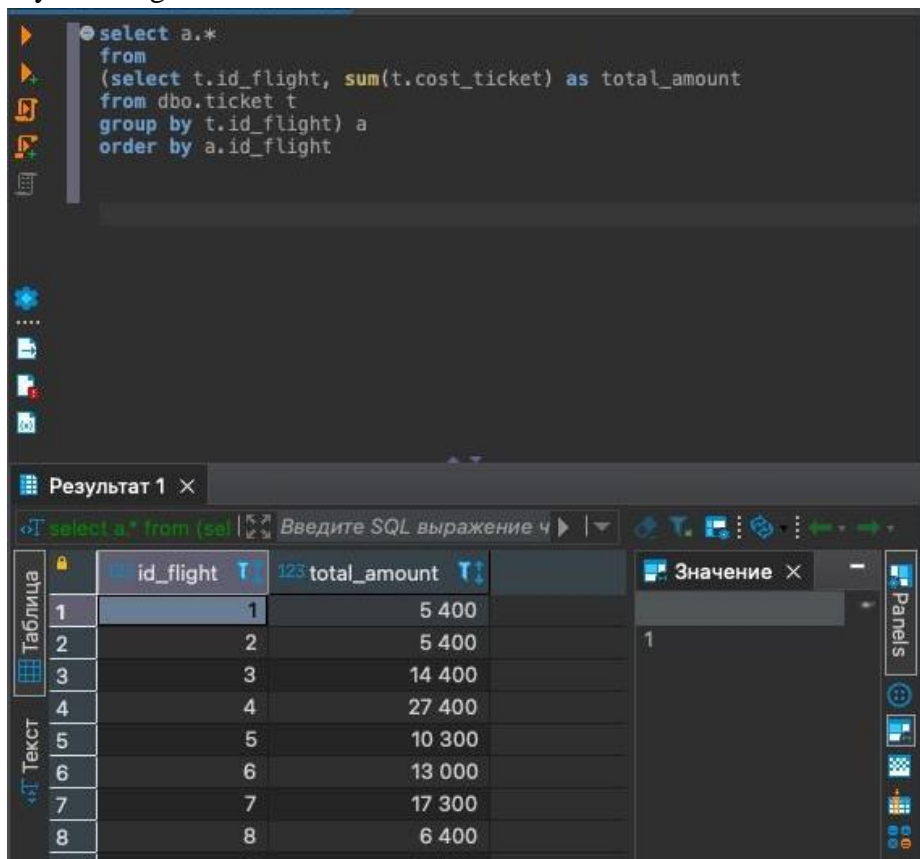
select a.* from (select r.depature_id_airport | Вве, ▶ | ▼

	depature_id_airport	last_dep
1	3	2021-07-04 10:00:00.000
2	7	2021-03-04 10:00:00.000
3	12	2021-04-04 10:00:00.000
4	15	2021-05-04 10:00:00.000
5	18	2021-06-04 10:00:00.000

Запрос №7

8)Итоговая сумма билетов, которые купили, в разрезе полетов (запрос с подзапросом в FROM, агрегированием, группировкой и сортировкой):

```
select a.*  
from (select t.id_flight, sum(t.cost_ticket) as total_amount  
from dbo.ticket t group by t.id_flight) a  
order by a.id_flight
```



The screenshot shows a SQL query window with the following text:

```
select a.*  
from  
(select t.id_flight, sum(t.cost_ticket) as total_amount  
from dbo.ticket t  
group by t.id_flight) a  
order by a.id_flight
```

Below the query window, the 'Results' pane displays the output of the query. The results are shown in a table with two columns: 'id_flight' and 'total_amount'. The table has 8 rows of data.

id_flight	total_amount
1	5 400
2	5 400
3	14 400
4	27 400
5	10 300
6	13 000
7	17 300
8	6 400

Запрос №8

9)Вывод полета, находящегося в ожидании, а также следующего за ним (запрос, использующий оконную функцию LEAD):

```
select f.id_flight, lead(f.id_flight) over (order by f.depature_date)  
from dbo.flight f where f.status_flight = 'В ожидании'
```

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

```
select f.id_flight, lead(f.id_flight) over (order by f.depature_date)
from dbo.flight f
where f.status_flight = 'В ожидании'
```

Below the query editor, the results are displayed in a table grid. The table has two columns: 'id_flight' and a second column representing the lead value. The results are as follows:

id_flight	Lead Value
2	5
5	8
8	11
11	14
14	[NULL]

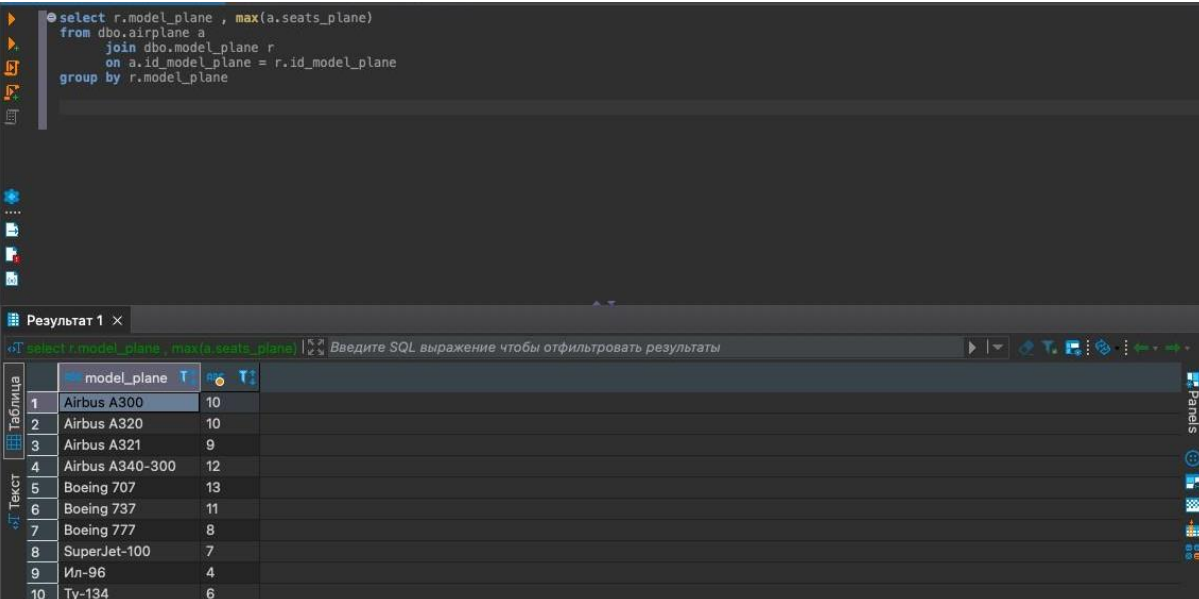
Запрос №9

10) Вывод количества мест для каждой модели самолета (запрос с агрегированием и выражением JOIN, включающим не менее 2 таблиц):

```
select r.model_plane ,
max(a.seats_plane) from dbo.airplane a
join dbo.model_plane r
on a.id_model_plane =
```

r.id_model_plane group by r.model_plane

Запрос №10



The screenshot shows a SQL query editor with a query window at the top and a results window at the bottom. The query window contains the following SQL code:

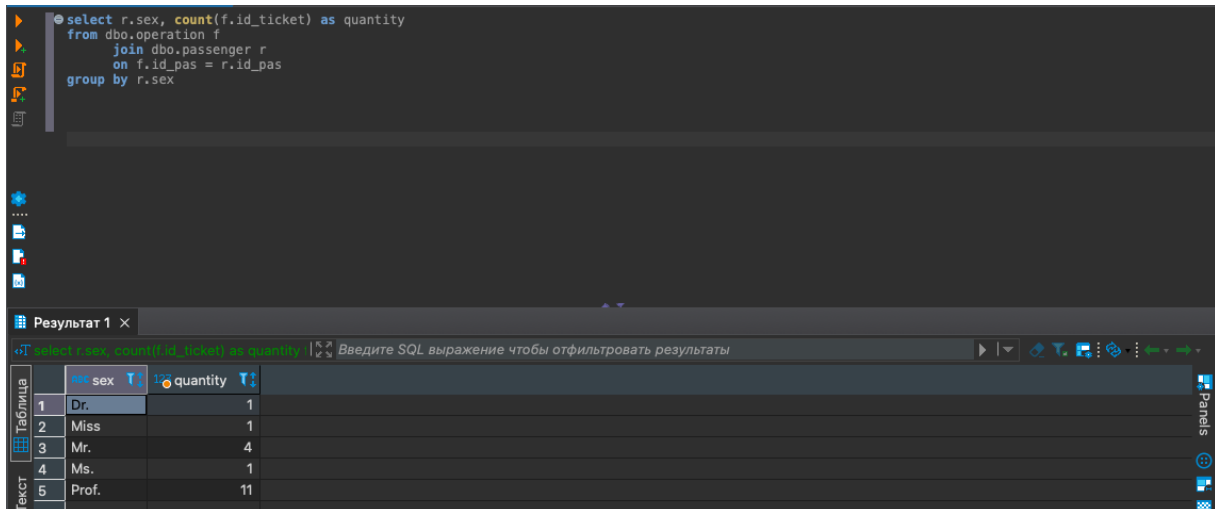
```
select r.model_plane , max(a.seats_plane)
from dbo.airplane a
join dbo.model_plane r
on a.id_model_plane = r.id_model_plane
group by r.model_plane
```

The results window, titled "Результат 1", displays the results of the query in a table format. The table has two columns: "model_plane" and "seats_plane". The results are as follows:

	model_plane	seats_plane
1	Airbus A300	10
2	Airbus A320	10
3	Airbus A321	9
4	Airbus A340-300	12
5	Boeing 707	13
6	Boeing 737	11
7	Boeing 777	8
8	SuperJet-100	7
9	Ил-96	4
10	Ту-134	6

11) Вывод количества купленных билетов по полу/статусу человека (запрос с агрегированием и выражением JOIN, включающим не менее 2 таблиц):

```
select r.sex, count(f.id_ticket) as  
quantity from dbo.operation f  
    join dbo.passenger r  
    on f.id_pas =  
    r.id_pas  
group by r.sex
```



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

```
select r.sex, count(f.id_ticket) as quantity  
from dbo.operation f  
    join dbo.passenger r  
    on f.id_pas = r.id_pas  
group by r.sex
```

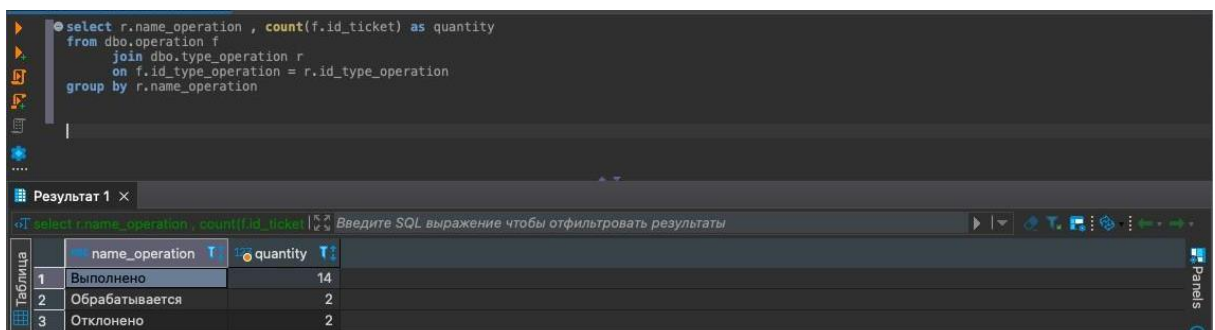
The results pane shows the following data:

sex	quantity
Dr.	1
Miss	1
Mr.	4
Ms.	1
Prof.	11

Запрос №11

12) Вывод количества билетов на каждой стадии обработки (запрос с агрегированием и выражением JOIN, включающим не менее 2 таблиц):

```
select r.name_operation , count(f.id_ticket) as  
quantity from dbo.operation f  
    join dbo.type_operation r  
    on f.id_type_operation =  
    r.id_type_operation group by r.name_operation
```



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

```
select r.name_operation , count(f.id_ticket) as quantity  
from dbo.operation f  
    join dbo.type_operation r  
    on f.id_type_operation = r.id_type_operation  
group by r.name_operation
```

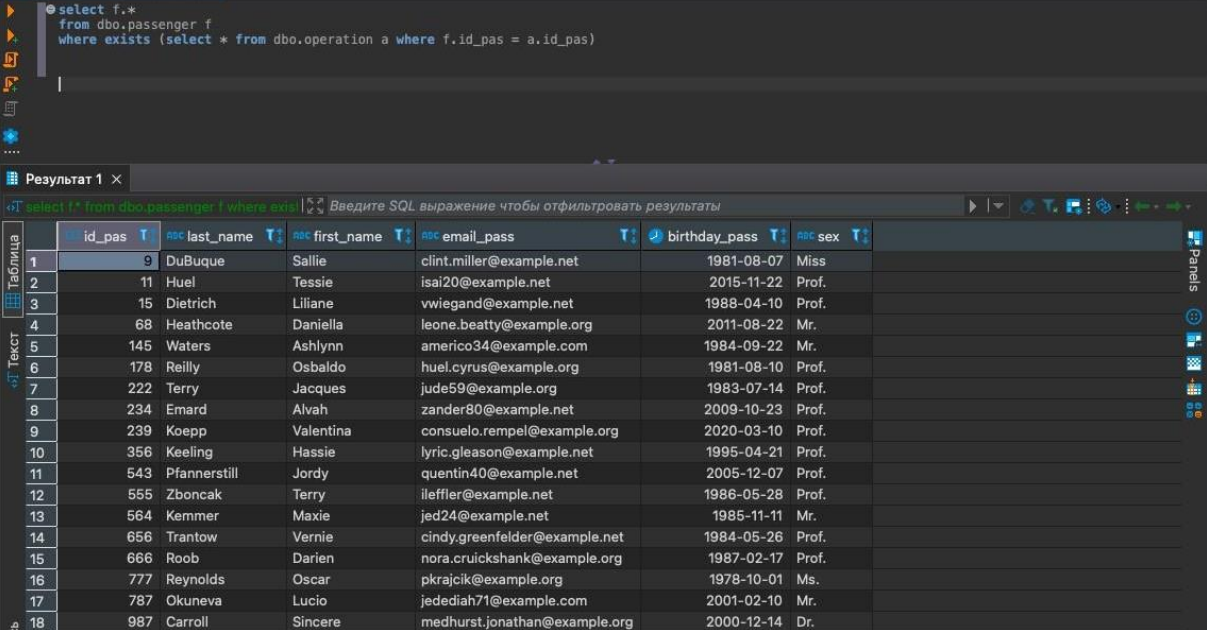
The results pane shows the following data:

name_operation	quantity
Выполнено	14
Обрабатывается	2
Отклонено	2

Запрос №12

13) Вывод данных о пассажирах, которые совершили любую операцию (запрос с EXISTS):

```
select f.* from dbo.passenger f
where exists (select * from dbo.operation a where f.id_pas = a.id_pas)
```

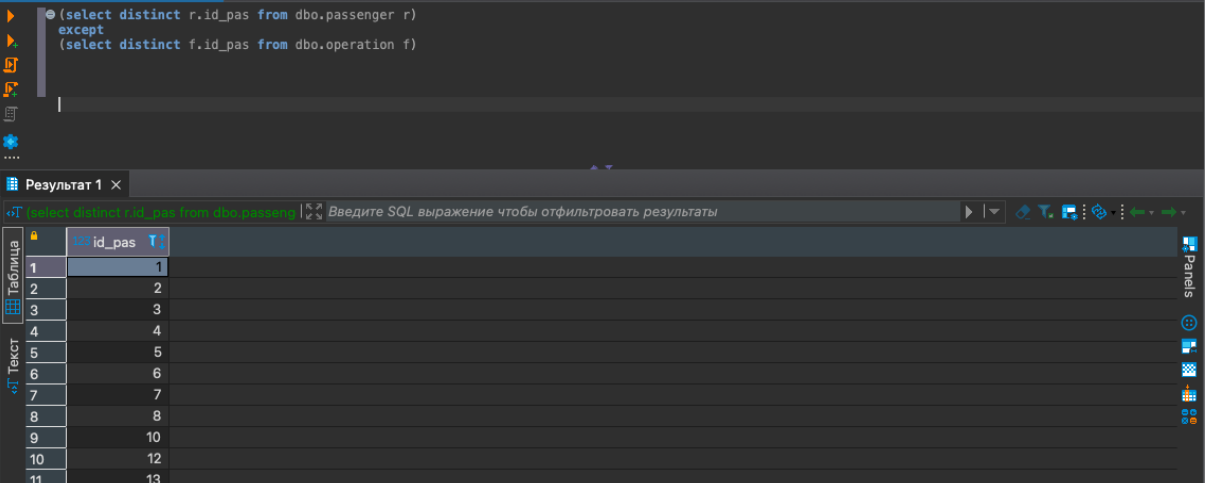


	id_pas	last_name	first_name	email_pass	birthday_pass	sex	
1	9	DuBuque	Sallie	clint.miller@example.net	1981-08-07	Miss	
2	11	Huel	Tessie	isai20@example.net	2015-11-22	Prof.	
3	15	Dietrich	Liliane	vwiegand@example.net	1988-04-10	Prof.	
4	68	Heathcote	Daniella	leone.beatty@example.org	2011-08-22	Mr.	
5	145	Waters	Ashlynn	americo34@example.com	1984-09-22	Mr.	
6	178	Reilly	Osbaldo	huel.cyrus@example.org	1981-08-10	Prof.	
7	222	Terry	Jacques	jude59@example.org	1983-07-14	Prof.	
8	234	Emard	Alvah	zander80@example.net	2009-10-23	Prof.	
9	239	Koepp	Valentina	consuelo.rempel@example.org	2020-03-10	Prof.	
10	356	Keeling	Hassie	lyric.gleason@example.net	1995-04-21	Prof.	
11	543	Pfannerstill	Jordy	quentin40@example.net	2005-12-07	Prof.	
12	555	Zboncak	Terry	ileffier@example.net	1986-05-28	Prof.	
13	564	Kemmer	Maxie	jed24@example.net	1985-11-11	Mr.	
14	666	Trantow	Vernie	cindy.greenfelder@example.net	1984-05-26	Prof.	
15	666	Roob	Darien	nora.cruickshank@example.org	1987-02-17	Prof.	
16	777	Reynolds	Oscar	pkrajcik@example.org	1978-10-01	Ms.	
17	787	Okuneva	Lucio	jedediah71@example.com	2001-02-10	Mr.	
18	987	Carroll	Sincere	medhurst.jonathan@example.org	2000-12-14	Dr.	

Запрос №13

14) Вывод ID клиентов, которые не совершали никаких операций (запрос, использующий манипуляции с множествами):

```
(select distinct r.id_pas from dbo.passenger r)
except
(select distinct f.id_pas from dbo.operation f)
```

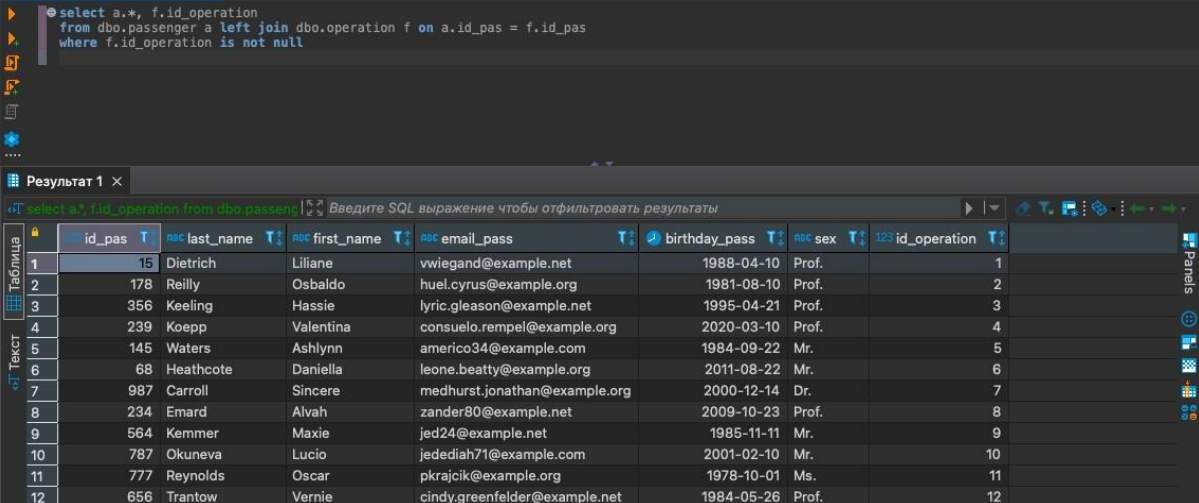


	id_pas
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	10
10	12
11	13

Запрос №14

15) Вывод полной информации по клиенту, который совершил хотя бы одну операцию, с присоединением ID совершенной операции (запрос с внешним соединением и проверкой на наличие NULL):

```
select a.*, f.id_operation
from dbo.passenger a left join dbo.operation f on a.id_pas = f.id_pas
where f.id_operation is not null
```



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

```
select a.*, f.id_operation
from dbo.passenger a left join dbo.operation f on a.id_pas = f.id_pas
where f.id_operation is not null
```

The result is displayed in a table grid with 12 rows and 8 columns. The columns are: id_pas, last_name, first_name, email_pass, birthday_pass, sex, and id_operation. The data is as follows:

	id_pas	last_name	first_name	email_pass	birthday_pass	sex	id_operation
1	15	Dietrich	Liliane	vwiegand@example.net	1988-04-10	Prof.	1
2	178	Reilly	Osbaldo	huel.cyrus@example.org	1981-08-10	Prof.	2
3	356	Keeling	Hassie	lyric.gleason@example.net	1995-04-21	Prof.	3
4	239	Koepp	Valentina	consuelo.rempel@example.org	2020-03-10	Prof.	4
5	145	Waters	Ashlynn	americo34@example.com	1984-09-22	Mr.	5
6	68	Heathcote	Daniella	leone.beatty@example.org	2011-08-22	Mr.	6
7	987	Carroll	Sincere	medhurst.jonathan@example.org	2000-12-14	Dr.	7
8	234	Emard	Alvah	zander80@example.net	2009-10-23	Prof.	8
9	564	Kemmer	Maxie	jed24@example.net	1985-11-11	Mr.	9
10	787	Okuneva	Lucio	jedediah71@example.com	2001-02-10	Mr.	10
11	777	Reynolds	Oscar	pkrajcik@example.org	1978-10-01	Ms.	11
12	656	Trantow	Vernie	cindy.greenfelder@example.net	1984-05-26	Prof.	12

Запрос №15

16) Вывод суммарной стоимости билетов каждого класса на каждый рейс (запрос с агрегированием и выражением JOIN, включающим не менее 3 таблиц/выражений):

```
select r.id_flight, a.id_class ,sum(t.cost_ticket) as quantity
from dbo.flight r join dbo.ticket t on r.id_flight =
t.id_flight
join dbo.places a on t.id_place = a.id_place
group by r.id_flight, a.id_class
```

```

select r.id_flight, a.id_class, sum(t.cost_ticket) as quantity
from dbo.flight r join dbo.ticket t on r.id_flight = t.id_flight
join dbo.places a on t.id_place = a.id_place
group by r.id_flight, a.id_class

```

	id_flight	id_class	quantity
12	12	1	4 500
13	13	1	5 000
14	3	2	12 000
15	5	2	10 000
16	6	2	10 000
17	7	2	15 000
18	11	2	10 000
19	12	2	10 000
20	13	2	13 000
21	4	3	25 000
22	9	3	28 000

Запрос №16

17) Вывод ID рейса, если на него куплено меньше 3 билетов с указанием количества купленных билетов (запрос с HAVING и агрегированием):

```

select r.id_flight, count(r.id_ticket) as
quantity from dbo.ticket r group by r.id_flight
having count(r.id_ticket) < 3

```

```

select r.id_flight, count(r.id_ticket) as quantity
from dbo.ticket r group by r.id_flight
having count(r.id_ticket) < 3

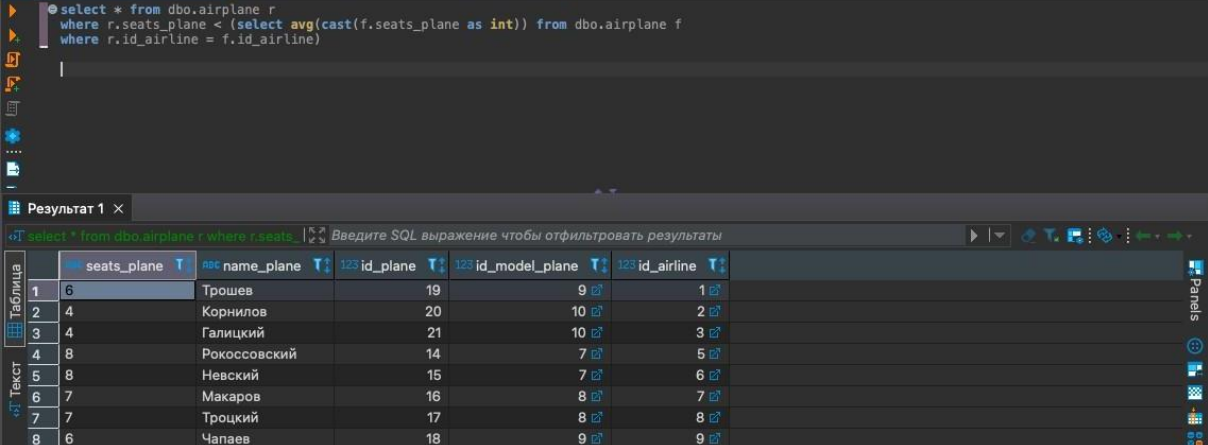
```

	id_flight	quantity
1	1	2
2	2	2
3	3	2
4	4	2
5	5	2
6	6	2
7	7	2
8	8	2

Запрос №17

18) Вывод информации о самолетах, у которых количество мест меньше среднего среди всех моделей с преобразованием типов данных (запрос с коррелированным подзапросом в WHERE):

```
select * from dbo.airplane r
where r.seats_plane < (select avg(cast(f.seats_plane as int)) from dbo.airplane f
where r.id_airline = f.id_airline)
```



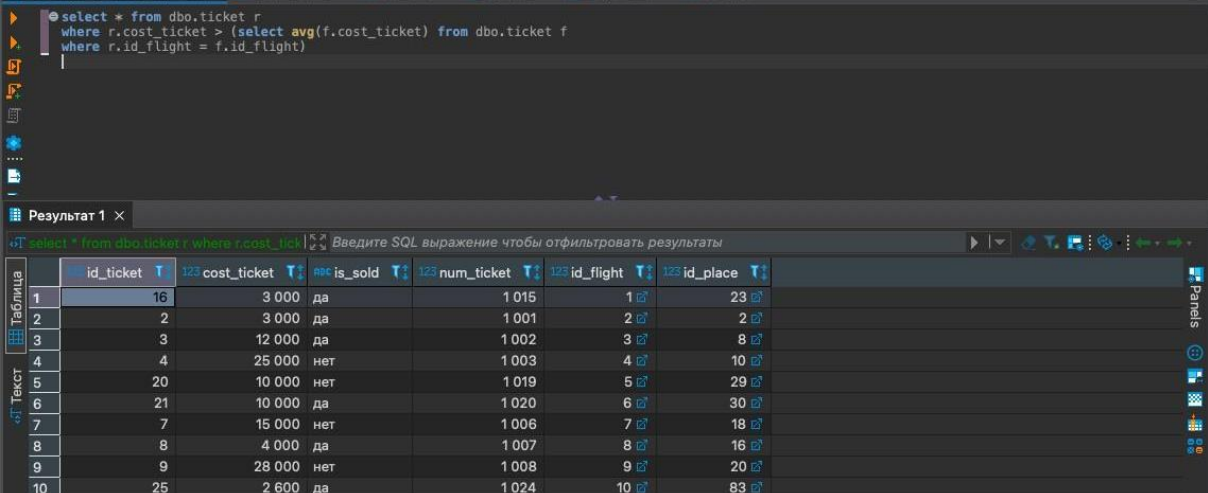
Результат 1

	seats_plane	name_plane	id_plane	id_model_plane	id_airline
1	6	Трошев	19	9	1
2	4	Корнилов	20	10	2
3	4	Галицкий	21	10	3
4	8	Рокоссовский	14	7	5
5	8	Невский	15	7	6
6	7	Макаров	16	8	7
7	7	Троцкий	17	8	8
8	6	Чапаев	18	9	9

Запрос №18

19) Вывод информации о билетах, у которых цена выше средней на конкретный рейс (запрос с коррелированным подзапросом в WHERE):

```
select * from dbo.ticket r
where r.cost_ticket > (select avg(f.cost_ticket) from dbo.ticket f
where r.id_flight = f.id_flight)
```



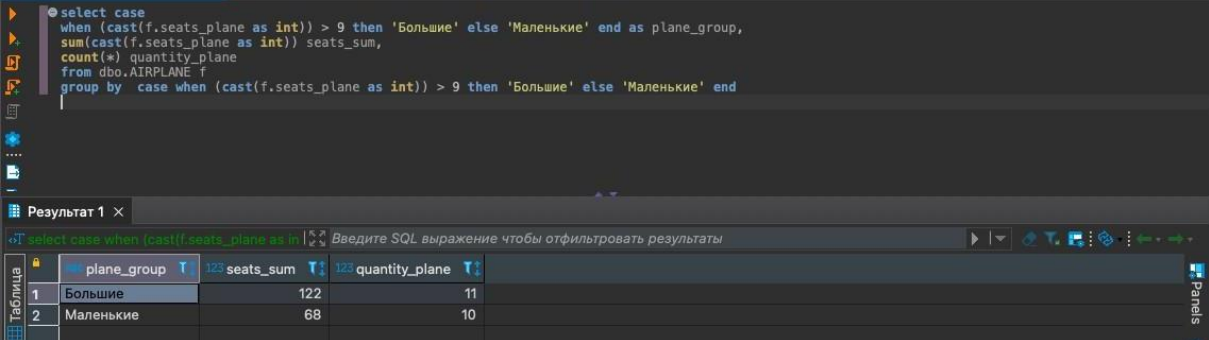
Результат 1

	id_ticket	cost_ticket	is_sold	num_ticket	id_flight	id_place
1	16	3 000	да	1 015	1	23
2	2	3 000	да	1 001	2	2
3	3	12 000	да	1 002	3	8
4	4	25 000	нет	1 003	4	10
5	20	10 000	нет	1 019	5	29
6	21	10 000	да	1 020	6	30
7	7	15 000	нет	1 006	7	18
8	8	4 000	да	1 007	8	16
9	9	28 000	нет	1 008	9	20
10	25	2 600	да	1 024	10	83

Запрос №19

20) Запрос разделяет самолеты на 2 группы: те, у кого больше 9 мест в салоне - большие, а остальные маленькие, выводит информацию о количестве самолетов в этих группах, а также сумму мест в каждой группе (запрос с CASE (IIF) и агрегированием):

```
select case
when (cast(f.seats_plane as int)) > 9 then 'Большие' else 'Маленькие' end as plane_group,
sum(cast(f.seats_plane as int)) seats_sum,
count(*) quantity_plane
from dbo.AIRPLANE f
group by case when (cast(f.seats_plane as int)) > 9 then 'Большие' else 'Маленькие' end
```



Результат 1

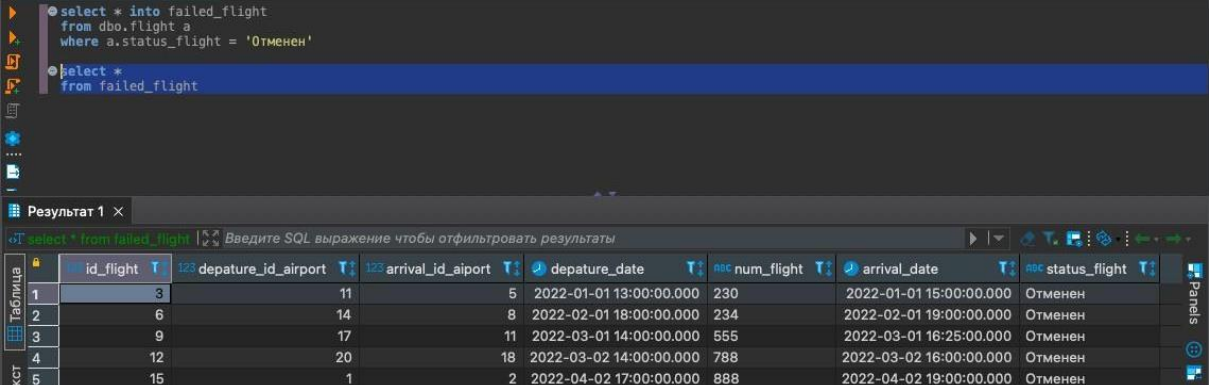
plane_group	seats_sum	quantity_plane
1 Большие	122	11
2 Маленькие	68	10

Запрос №20

21) Запрос создает таблицу failed_flight и вносит в нее отмененные рейсы из таблицы FLIGHT, сохраняя колонки и их типы данных (запрос SELECT INTO для подготовки выгрузки):

```
select * into failed_flight
from dbo.flight a
where a.status_flight = 'Отменен'
```

```
select *
from failed_flight
```



Результат 1

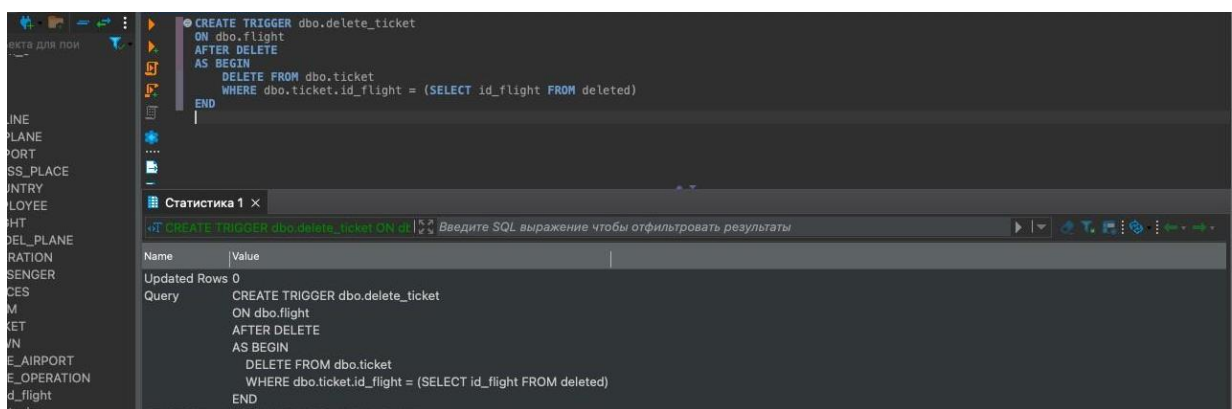
id_flight	departure_id_airport	arrival_id_airport	departure_date	num_flight	arrival_date	status_flight
3	11	5	2022-01-01 13:00:00.000	230	2022-01-01 15:00:00.000	Отменен
6	14	8	2022-02-01 18:00:00.000	234	2022-02-01 19:00:00.000	Отменен
9	17	11	2022-03-01 14:00:00.000	555	2022-03-01 16:25:00.000	Отменен
12	20	18	2022-03-02 14:00:00.000	788	2022-03-02 16:00:00.000	Отменен
15	1	2	2022-04-02 17:00:00.000	888	2022-04-02 19:00:00.000	Отменен

Запрос №21

Разработка триггера

Для оптимизации процесса хранения данных был создан триггер, который при удалении полета в таблице FLIGHT удаляет все билеты на данный полет в таблице TICKET:

```
CREATE TRIGGER
dbo.delete_ticket ON dbo.flight
AFTER DELETE
AS BEGIN
    DELETE FROM dbo.ticket
    WHERE dbo.ticket.id_flight = (SELECT id_flight FROM deleted)
END
```



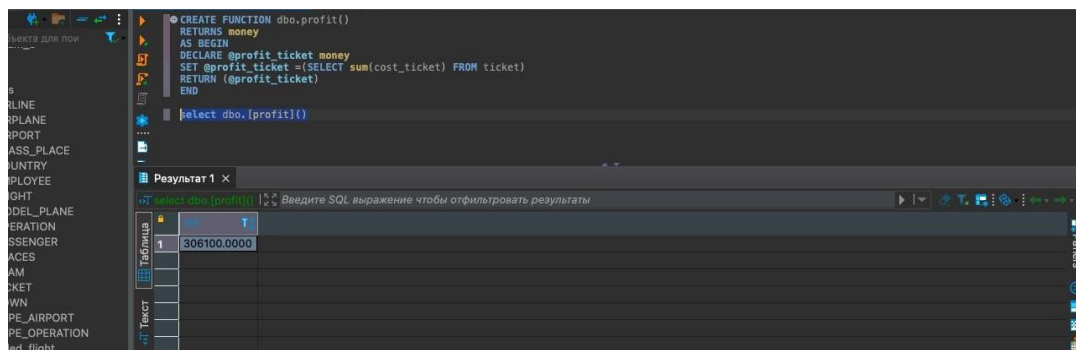
Демонстрация триггера

Разработка функций

1) Функция рассчитывает прибыль, полученную за продажу всех билетов:

```
CREATE FUNCTION
dbo.profit() RETURNS money
AS BEGIN
    DECLARE @profit_ticket money
    SET @profit_ticket =(SELECT sum(cost_ticket) FROM ticket)
    RETURN (@profit_ticket)
END
```

```
select dbo.[profit]()
```



Демонстрация функции №1

2) Функция определяет количество полетов, которые имеют статус “В ожидании”: CREATE FUNCTION dbo.amount_flight()

RETURNS int

AS BEGIN

DECLARE @amt_flight int

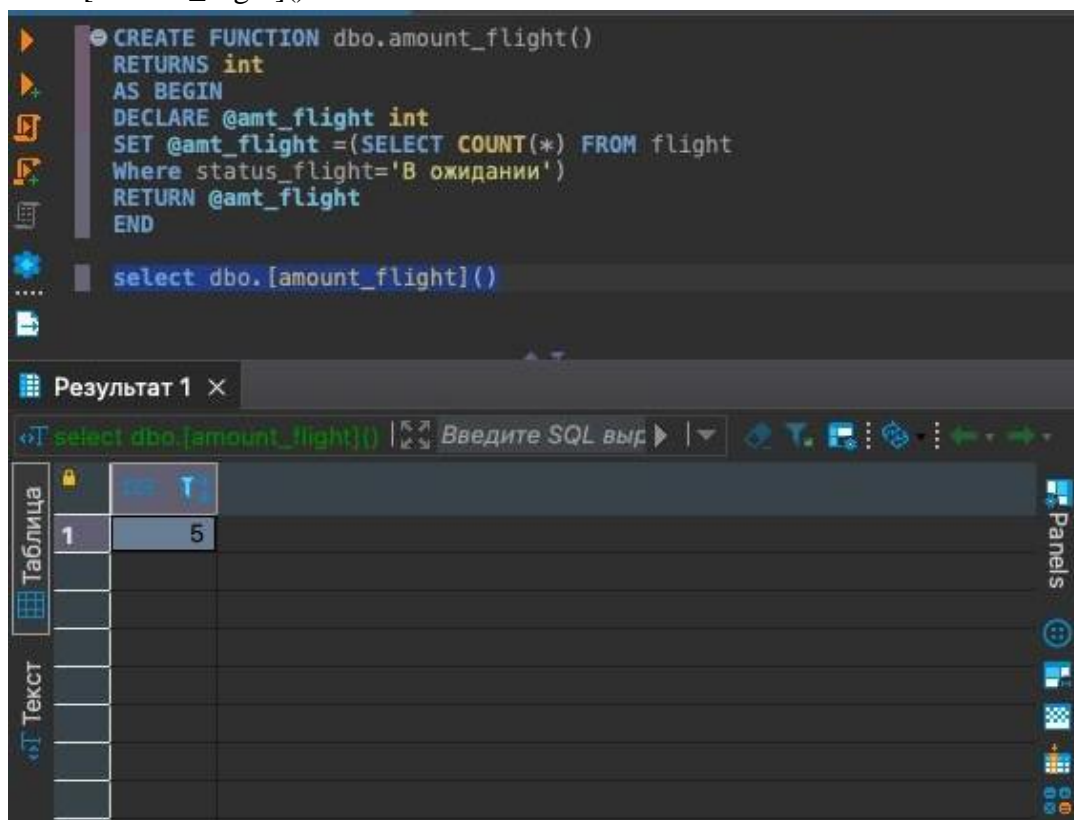
SET @amt_flight =(SELECT COUNT(*) FROM flight

Where status_flight='В ожидании')

RETURN @amt_flight

END

select dbo.[amount_flight]()



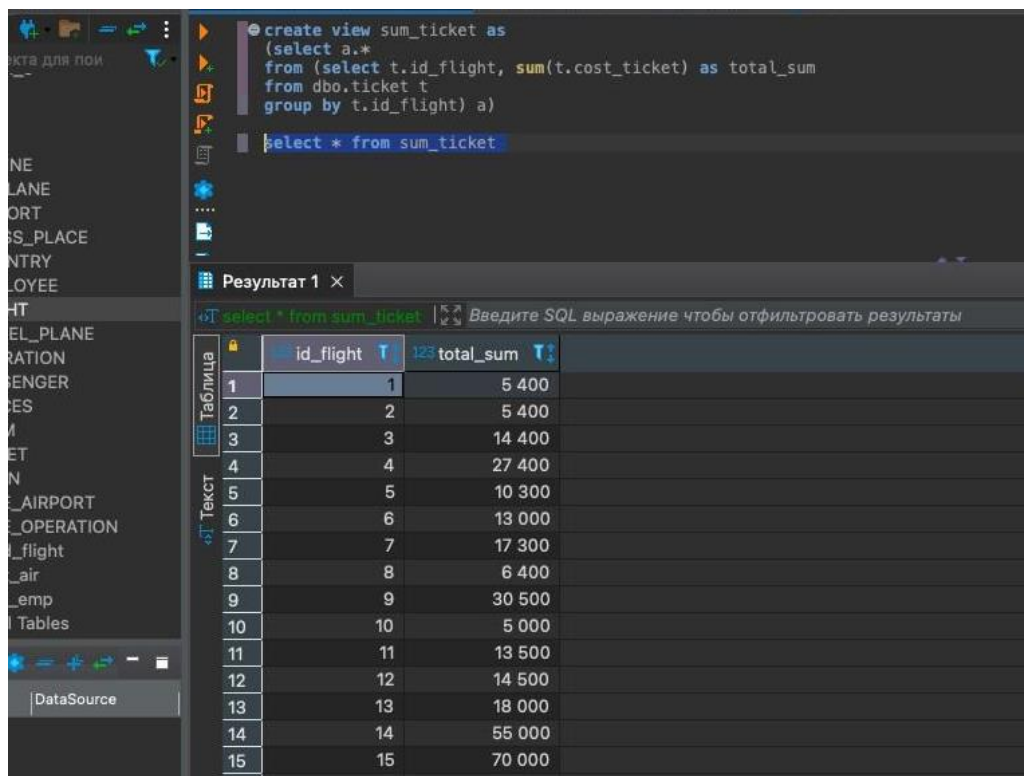
Демонстрация функции №2

Разработка представлений

1) Представление, показывающее сумму проданных билетов по каждому из полетов:

```
create view sum_ticket as
(select a.*
from (select t.id_flight, sum(t.cost_ticket) as total_sum
from dbo.ticket t
group by t.id_flight) a)

select * from sum_ticket
```



Результат 1

	id_flight	total_sum
1	1	5 400
2	2	5 400
3	3	14 400
4	4	27 400
5	5	10 300
6	6	13 000
7	7	17 300
8	8	6 400
9	9	30 500
10	10	5 000
11	11	13 500
12	12	14 500
13	13	18 000
14	14	55 000
15	15	70 000

Демонстрация представления №1

2) Представление, показывающее последний прилёт в каждый аэропорт на данный момент:

```
create view flight_last as
(select a.*
from (select f.arrival_id_aiport, max(f.arrival_date) as last_flight
from dbo.flight f
where f.status_flight =
'Окончен' group by
f.arrival_id_aiport) a )

select * from flight_last
```


Скриншот SQL Server Enterprise Manager. В центре экрана отображен код SQL-запроса:

```

create view flight_last as
(select a.*
 from (select f.arrival_id_aiport, max(f.arrival_date) as last_flight
 from dbo.flight f
 where f.status_flight = 'Окончен'
 group by f.arrival_id_aiport) a )

select * from flight_last
  
```

В нижней части экрана отображены результаты запроса в виде таблицы:

Таблица	arrival_id_aiport	last_flight
1	1	2021-06-04 12:00:00.000
2	3	2021-03-04 11:00:00.000
3	6	2021-04-04 12:00:00.000
4	9	2021-05-04 11:30:00.000
5	18	2021-07-04 13:00:00.000

Демонстрация представления №2

Разработка процедур

1) Процедура, удаляющая данные об отмененном полете:

```

CREATE PROCEDURE
del_failed_flight @id int
AS
DELETE FROM failed_flight
WHERE id_flight = @id
  
```

Скриншот SQL Server Enterprise Manager. В центре экрана отображен код SQL-запроса:

```

CREATE PROCEDURE del_failed_flight
@id int
AS
DELETE FROM failed_flight
WHERE id_flight = @id
  
```

В нижней части экрана отображены статистические данные о выполнении запроса:

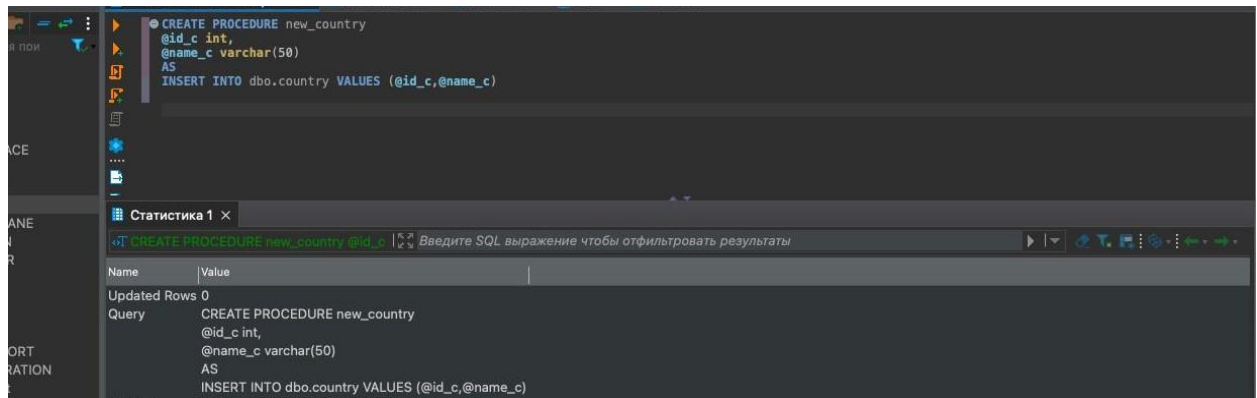
Имя	Значение
Updated Rows	0
Query	CREATE PROCEDURE del_failed_flight @id int AS DELETE FROM failed_flight WHERE id_flight = @id

Демонстрация процедуры №1

2) Процедура, добавляющая новую страну в таблицу

COUNTRY:

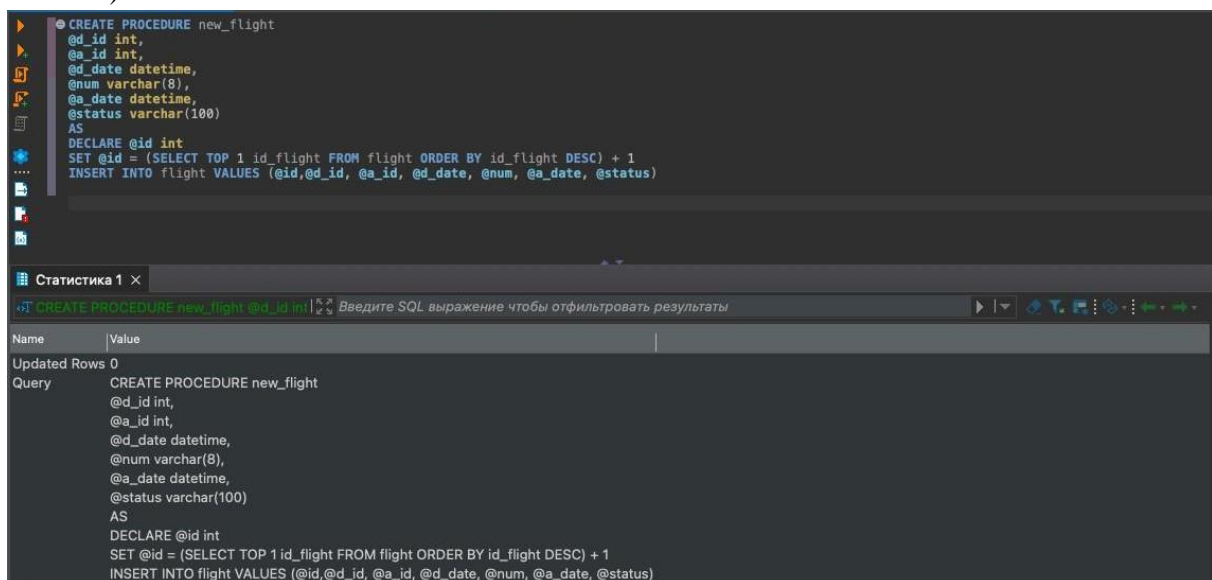
```
CREATE PROCEDURE
new_country @id_c int,
@name_c varchar(50)
AS
INSERT INTO dbo.country VALUES (@id_c,@name_c)
```



Демонстрация процедуры №2

3) Процедура, добавляющая новый полет в таблицу FLIGHT:

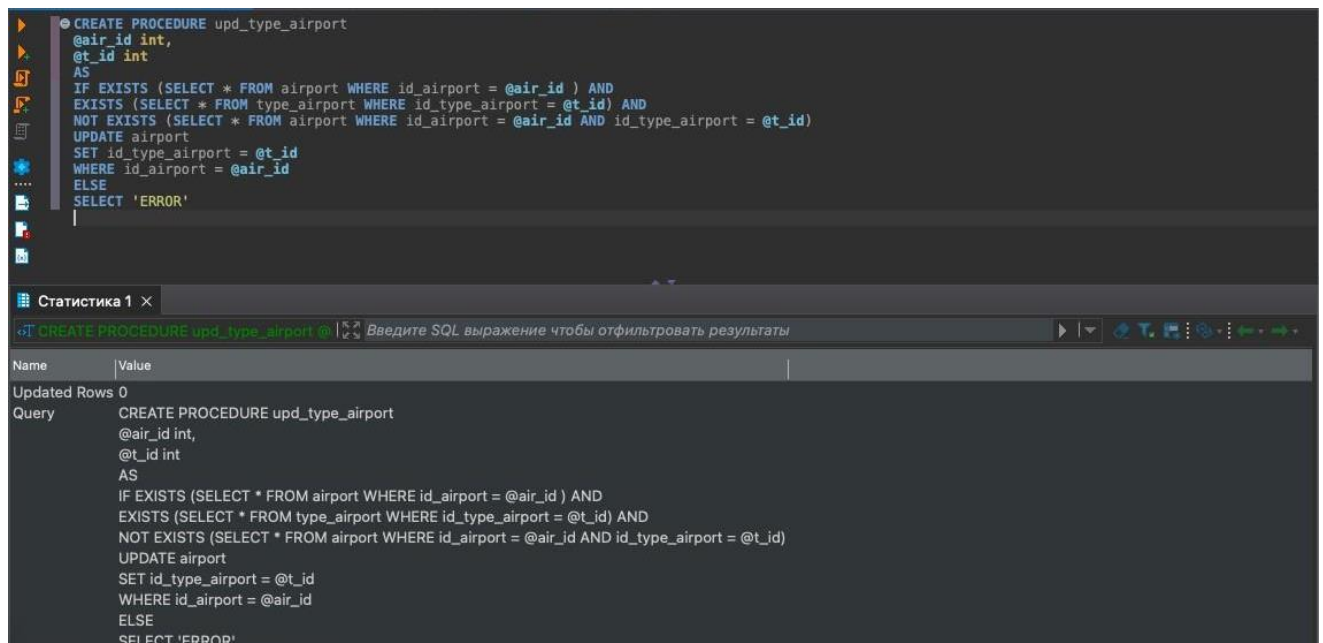
```
CREATE PROCEDURE
new_flight @d_id int,
@a_id int,
@d_date datetime,
@num varchar(8),
@a_date datetime,
@status varchar(100)
AS
DECLARE @id int
SET @id = (SELECT TOP 1 id_flight FROM flight ORDER BY id_flight DESC) + 1
INSERT INTO flight VALUES (@id, @d_id, @a_id, @d_date, @num, @a_date,
@status)
```



Демонстрация процедуры №3

4)Процедура, обновляющая тип аэропорта:

```
CREATE PROCEDURE
upd_type_airport @air_id int,
@t_id int
AS
IF EXISTS (SELECT * FROM airport WHERE id_airport = @air_id ) AND
EXISTS (SELECT * FROM type_airport WHERE id_type_airport = @t_id) AND
NOT EXISTS (SELECT * FROM airport WHERE id_airport = @air_id AND
id_type_airport = @t_id)
UPDATE airport
SET id_type_airport = @t_id
WHERE id_airport = @air_id
ELSE
SELECT 'ERROR'
```



Демонстрация процедуры №4

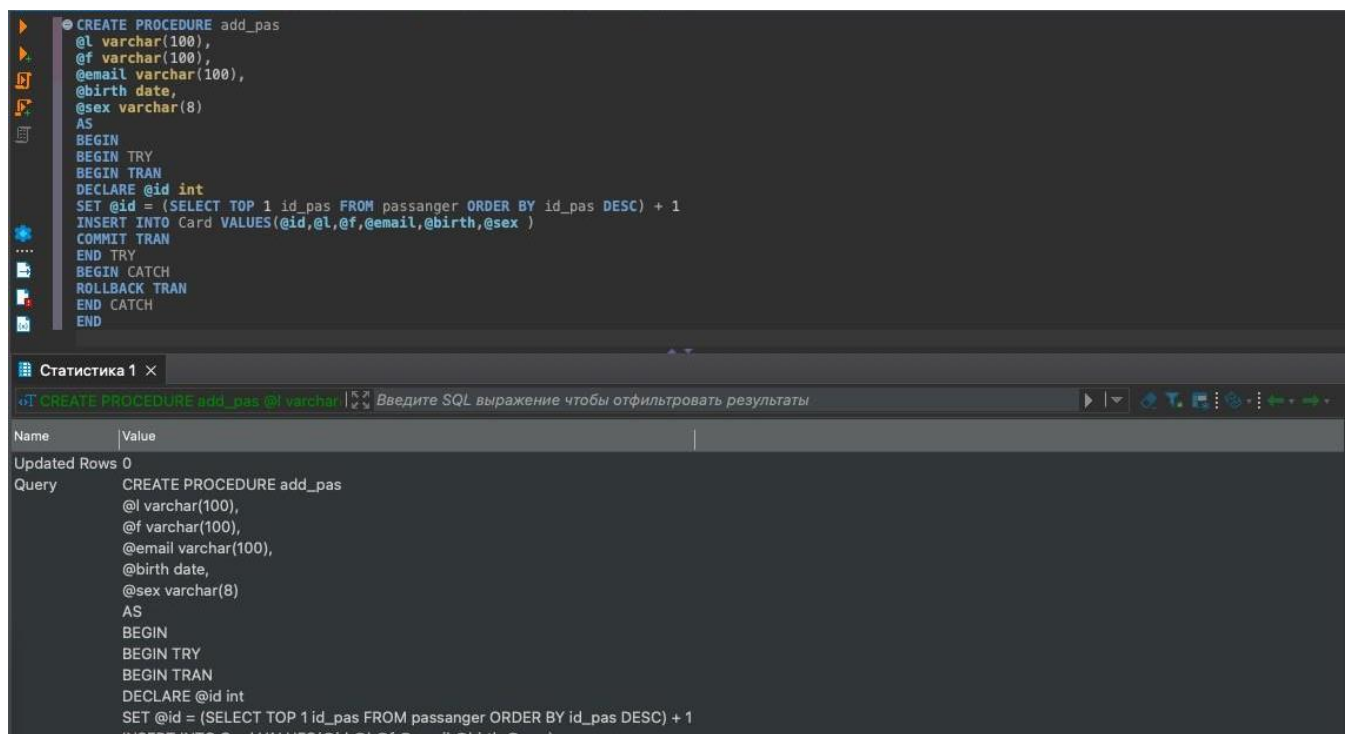
5)Процедура, добавляющая пассажира и его данные в таблицу PASSENGER:

```
CREATE PROCEDURE
add_pas @l varchar(100),
@f varchar(100),
@email varchar(100),
@birth date,
@sex varchar(8)
AS
BEGIN
BEGIN
TRY
```

```

BEGIN TRAN
DECLARE @id int
SET @id = (SELECT TOP 1 id_pas FROM passanger ORDER BY id_pas DESC) +
1 INSERT INTO Card VALUES(@id,@l,@f,@email,@birth,@sex )
COMMIT TRAN
END TRY
BEGIN CATCH
ROLLBACK TRAN
END CATCH
END

```



Демонстрация процедуры №5

Информационная панель в Excel

Для создания информационной панели было решено выгрузить несколько таблиц с заполненными данными в Excel. Мы построили срез для таблицы PASSENGER, который может показать по полу/статусу информацию о пассажирах:

	A	B	C	D	E	F	G	H	I	J	K
1	id_pas	last_name	first_name	email_pass	birthday_pass	sex					
2	1	Ratke	Melba	btromp@example.org	27.07.1996	Miss					
3	2	Erdman	Larue	raegan.zieme@example.net	03.08.2003	Prof.					
4	3	Smitham	Westley	niko18@example.net	09.07.1994	Dr.					
5	4	Graham	Zaria	donavon81@example.org	15.07.1998	Dr.					
6	5	Altenwerth	Katheryn	ewald.morar@example.org	06.03.2013	Prof.					
7	6	Kling	Sigrid	caterina36@example.org	28.02.2022	Mrs.					
8	7	Rowe	Marques	jjast@example.net	04.07.2006	Miss					
9	8	Green	Erick	millie51@example.com	03.07.2019	Dr.					
10	9	DuBuque	Sallie	clint.miller@example.net	07.08.1981	Miss					
11	10	Will	Buster	mack.shields@example.net	06.07.2011	Mr.					
12	11	Huel	Tessie	isai20@example.net	22.11.2015	Prof.					
13	12	Gusikowski	Mathias	harmony87@example.com	23.01.1982	Dr.					
14	13	Bauch	Hayley	ckuvalis@example.com	07.07.1971	Prof.					
15	14	Hickle	Desiree	percy23@example.org	26.07.2018	Dr.					
16	15	Dietrich	Liliane	vwiegand@example.net	10.04.1988	Prof.					
17	16	Mosciski	Jeff	jalon.glover@example.com	03.06.1990	Dr.					
18	17	Schuppe	Luella	dosinski@example.net	04.07.1992	Mr.					
19	18	Conner	Libbie	tgutkowski@example.net	03.03.1977	Prof.					
20	19	Heathcote	Frederic	lennie.mckenzie@example.com	04.04.2000	Mr.					

sex

Dr.

Miss

Mr.

Mrs.

Ms.

Prof.

Срез для таблицы PASSENGER

Также построили срез для таблицы TICKET, который быстро может показать информацию о проданных и непроданных билетах:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	id_ticket	cost_ticket	is_sold	num_ticket	id_flight	id_place										
2	1	2400	да	1000	1	1										
3	2	3000	да	1001	2	2										
4	3	12000	да	1002	3	8										
5	4	25000	нет	1003	4	10										
6	5	300	да	1004	5	11										
7	6	3000	нет	1005	6	12										
8	7	15000	нет	1006	7	18										
9	8	4000	да	1007	8	16										
10	9	28000	нет	1008	9	20										
11	10	2400	да	1009	10	17										
12	11	3500	нет	1010	11	21										
13	12	4500	да	1011	12	22										
14	13	5000	нет	1012	13	23										
15	14	30000	да	1013	14	31										
16	15	35000	нет	1014	15	32										
17	16	3000	да	1015	1	23										
18	17	2400	нет	1016	2	25										
19	18	2400	да	1017	3	24										
20	19	2400	да	1018	4	26										
21	20	10000	нет	1019	5	29										
22	21	10000	да	1020	6	30										
23	22	2300	да	1021	7	41										
24	23	2400	да	1022	8	51										
25	24	2500	нет	1023	9	61										
26	25	2600	да	1024	10	83										
27	26	10000	да	1025	11	53										
28	27	10000	нет	1026	12	54										
29	28	13000	да	1027	13	70										
30	29	25000	да	1028	14	89										
31	30	35000	нет	1029	15	65										
32																

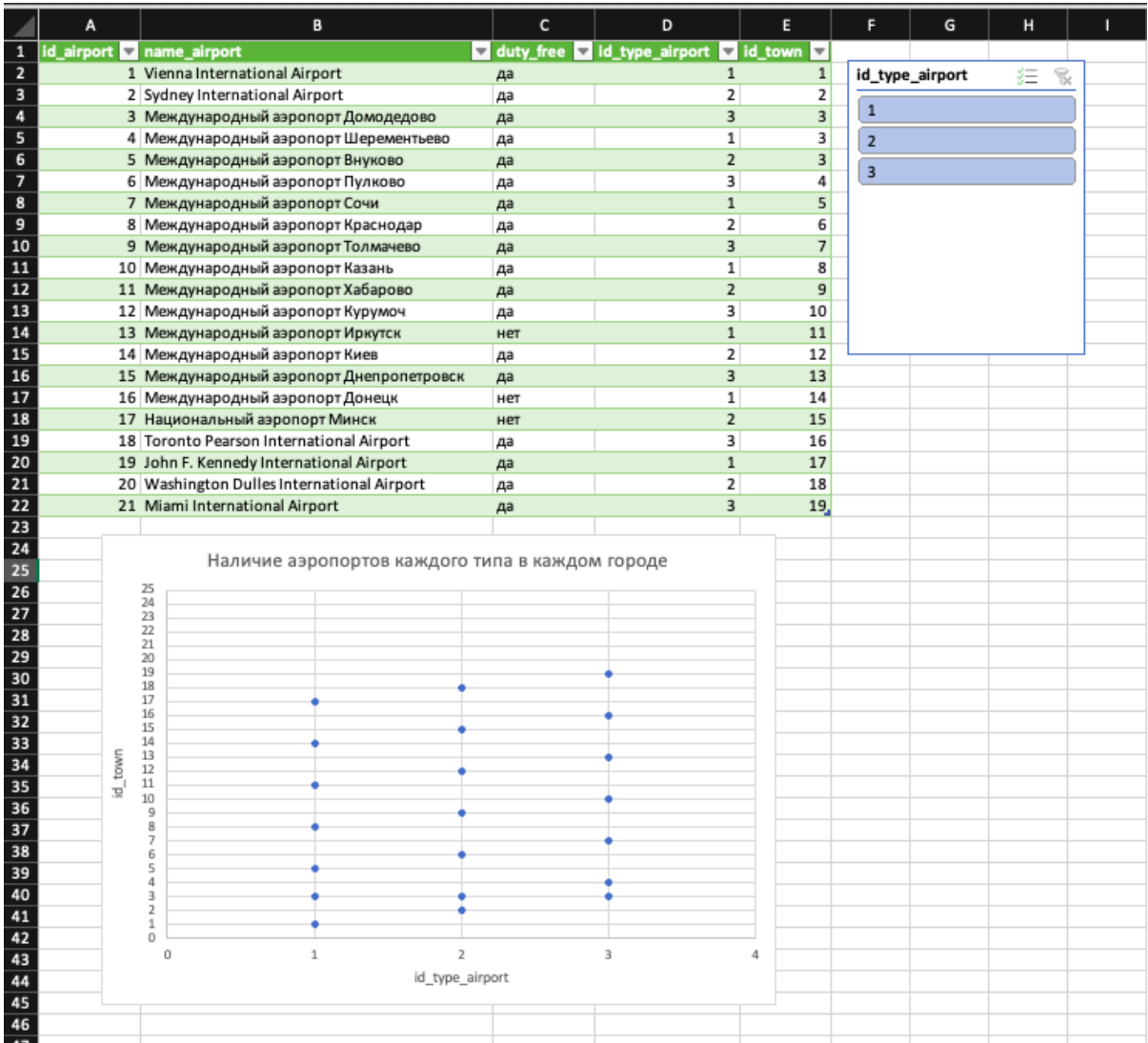
is_sold

да

нет

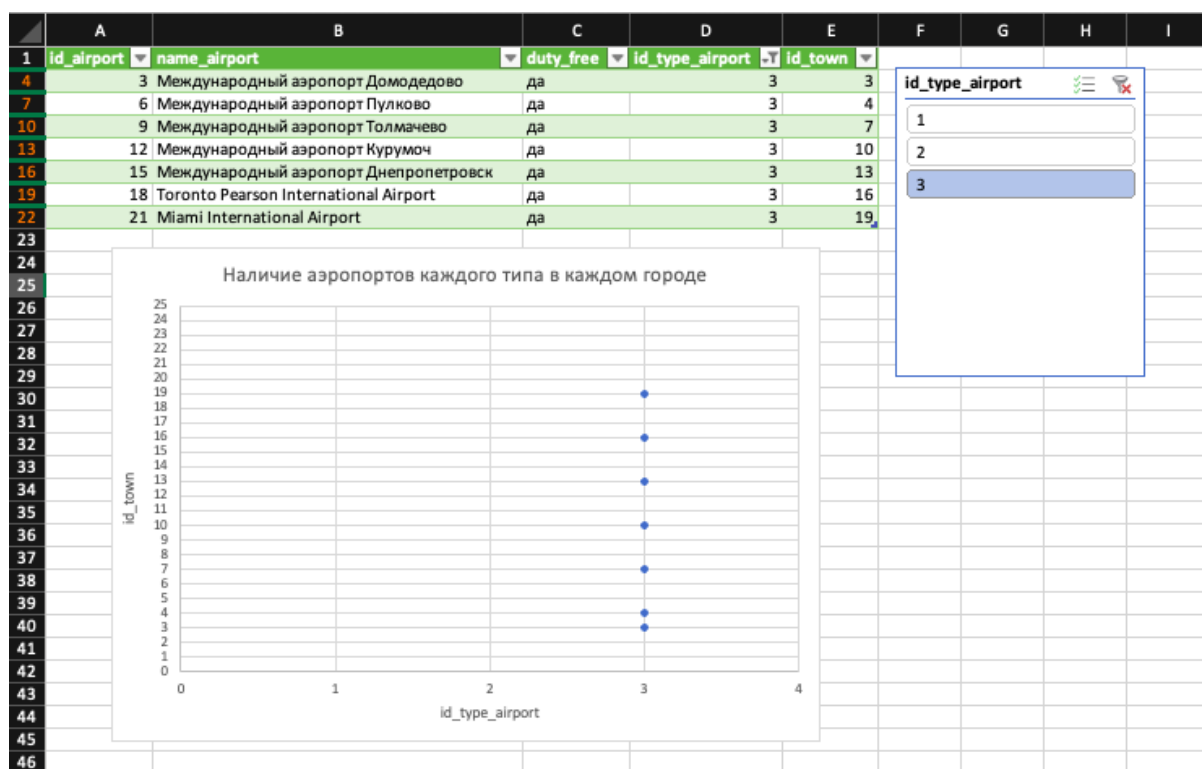
Срез для таблицы TICKET

После мы построили срез для таблицы AIRPORT, который выбирает аэропорты каждого типа, и построили точечный график, который показывает наличие аэропортов каждого типа в каждом из городов:



Срез и точечный график для таблицы AIRPORT

Также, при выборе определенного типа аэропорта в срезе, график обновляется:



Пример обновления графика

Отчеты, визуализация

Первый отчет:

В первом графике отображаются модели самолетов и максимальное количество мест в нем.

Во втором графике мы видим модель самолета и статус рейса. Был ли он выполнен или отменен. То есть простаивает ли он в определенном аэропорту или меняет свое местоположение

В последнем графике наглядно показано, в каком аэропорту какой самолет находится. Для нас это очень важно с той точки зрения, чтобы выстраивать последующие рейсы с этими моделями самолетов.

В данном отчете мы хотели демонстрируем, как связаны самолеты, статус рейса этих самолетов (отменен или выполнен), а также где он находится в данный момент.

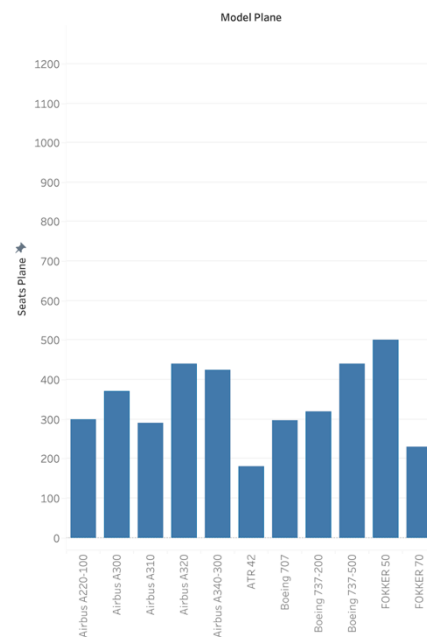
Статус рейса каждого самолета

Model Plane	Status Flight	
	Окончен	Отменен
Airbus A220-100	●	
Airbus A300	●	
Airbus A340-600	●	
Boeing 707 MAX 9		●
Boeing 737-800	●	

Местоположение каждого самолета

Name Airport	Model Plane				
	Airbus A220-100	Airbus A300	Airbus A340-600	Boeing 707 MAX..	Boeing 737-800
Abu Dhabi International A..	●				●
London Heathrow Airport			●		
Miami International Airpo..		●			
Paris-Charles-de-Gaulle Ai..				●	

Количество мест в самолете



Отчет 1

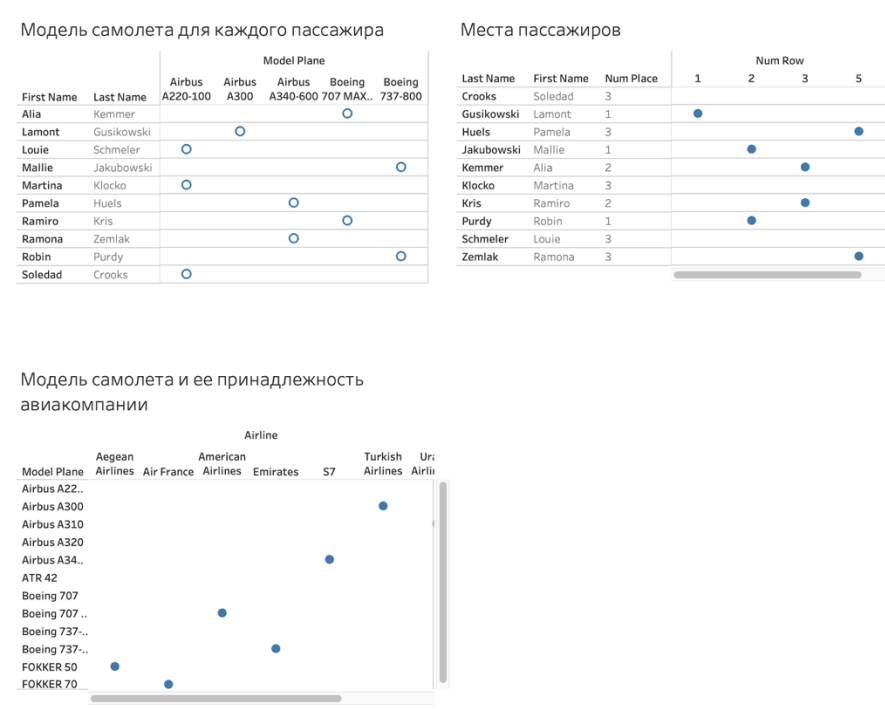
Второй отчет:

Во втором отчете мы можем наблюдать пассажиров, а конкретнее, имя и фамилию, купленные места в самолете и выбранную авиакомпанию.

В первой таблице выведены имя и фамилия пассажиров, купленное место, а точнее ряд и место (сказать, что в ряду 3 места).

На второй таблице можно увидеть, в какой модели самолета было приобретено место.

На третьей таблице изображено, какой авиакомпанией летит пассажир и соответственно какой авиакомпании принадлежит той или иной самолет.



Отчет 2

Третий отчет:

В данном отчете представлены данные о билетах и их статусе продаж.

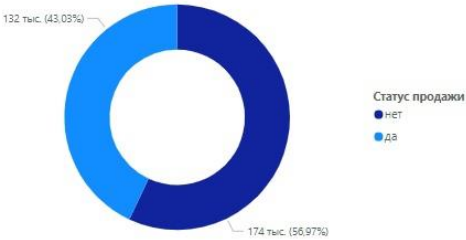
В первой таблице показаны имена и фамилии пассажиров, а также идентификаторы билетов, которые они приобрели. Это позволяет нам отслеживать, какие билеты были куплены каждым пассажиром.

На круговой диаграмме отображена общая стоимость всех билетов, разделенная по статусу продажи. Мы можем видеть, какая часть билетов была продана (статус "да") и какая часть осталась непроданной (статус "нет"). Диаграмма демонстрирует, что 56.97% билетов были проданы, а 43.03% остались непроданными, что позволяет нам оценить эффективность продаж.

В правой таблице представлена детализированная информация по каждому билету, включая их идентификаторы и стоимость. Таблица также разделена на две колонки, указывающие, был ли билет продан или нет, и общую стоимость билетов в каждой категории.

first_name	id_pas	last_name	id_ticket
Alvah	234	Emard	14
Ashlynn	145	Waters	8
Daniella	68	Heathcote	10
Darien	666	Roob	23
Hassie	356	Keeling	3
Jacques	222	Terry	26
Jordy	543	Pfannerstill	25
Liliane	15	Dietrich	1
Lucio	787	Okuneva	18
Maxie	564	Kemmer	16
Osbaldio	178	Reilly	2
Oscar	777	Reynolds	19
Sallie	9	DuBuque	29
Sincere	987	Carroll	12
Terry	555	Zboncak	22
Tessie	11	Huel	28
Valentina	239	Koepp	5
Vernie	656	Trantow	21

Общая стоимость всех билетов по Статус продажи



id_ticket	да	нет	Всего
1	2400		2400
2	3000		3000
3	12000		12000
4		25000	25000
5	300		300
6		3000	3000
7		15000	15000
8	4000		4000
9		28000	28000
10	2400		2400
11		3500	3500
12	4500		4500
13		5000	5000
14	30000		30000
15		35000	35000
16	3000		3000
17		2400	2400
18	2400		2400
19	2400		2400
20		10000	10000
21	10000		10000
22	2300		2300
23	2400		2400
24		2500	2500
25	2600		2600
26	10000		10000
27		10000	10000
28	13000		13000
29	25000		25000
30		35000	35000
Всего	131700	174400	306100