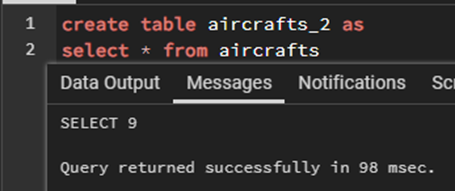
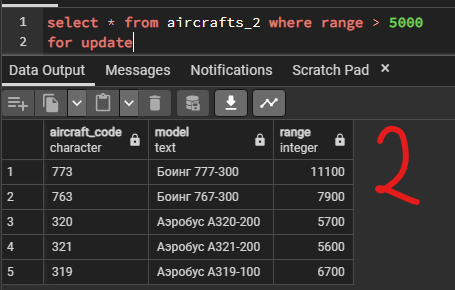
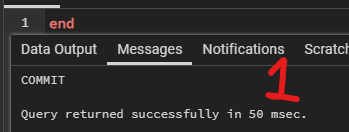
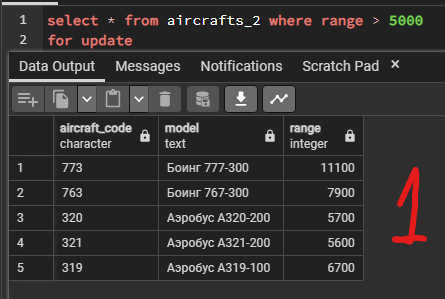
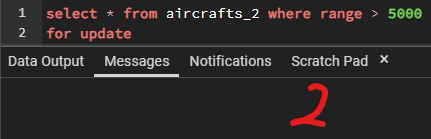
Букин Николай ПМ22-3

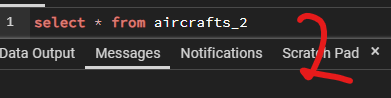
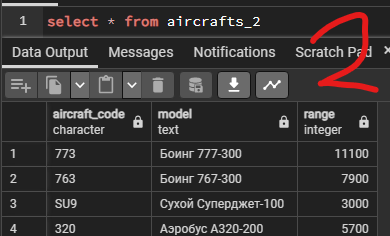
Лабораторная работа №5

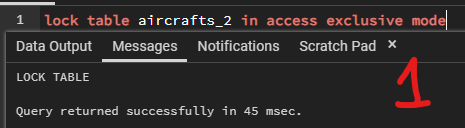
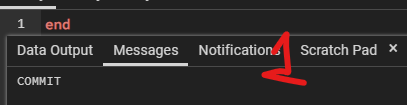
№1

1. 

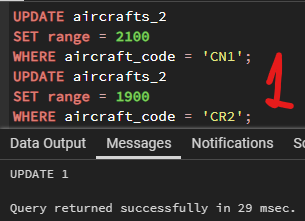
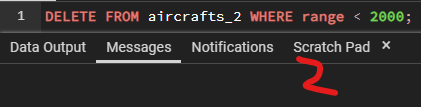
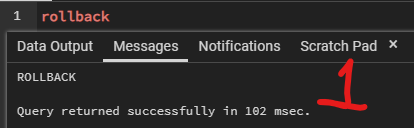
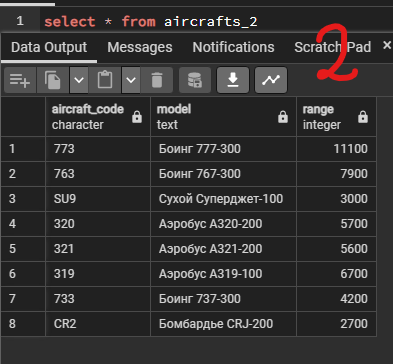
Создадим дубликат

1. ****





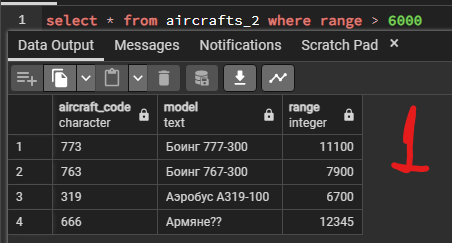
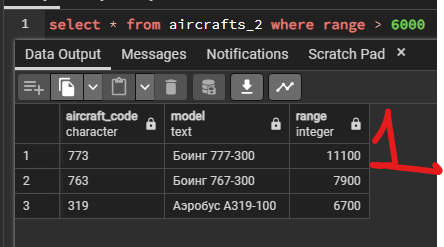
№2

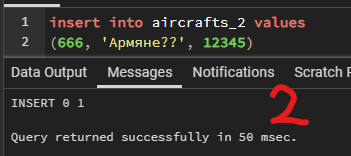
****

№3

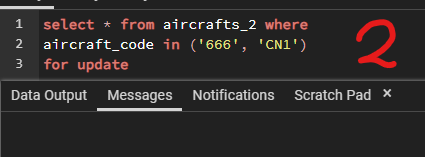
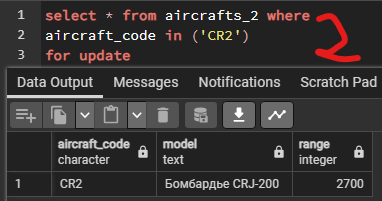
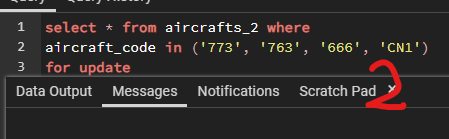
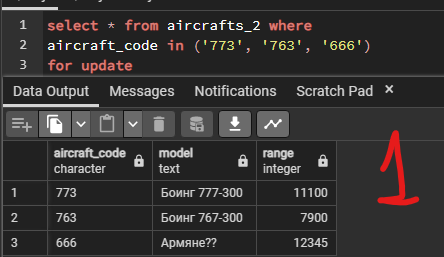
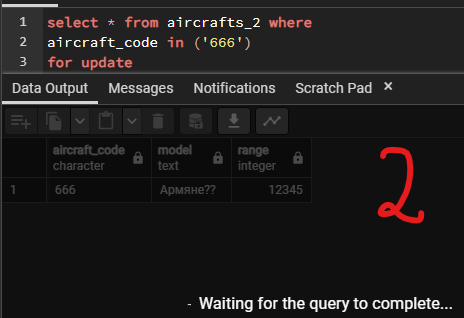
Потерянное обновление - это ситуация, когда несколько транзакций одновременно изменяют одни и те же данные, и все изменения, кроме последнего, теряются. В нашем случае значение 2100 было потеряно, а затем установлено последнее значение 2500. Это означает, что обновление данных было потеряно. Чтобы предотвратить такие ситуации, можно обновлять значения последовательно, сначала фиксируя изменения 2100, а затем 2500.

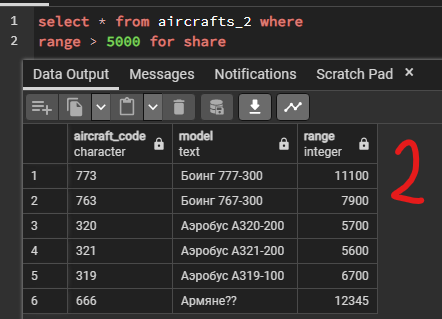
№4

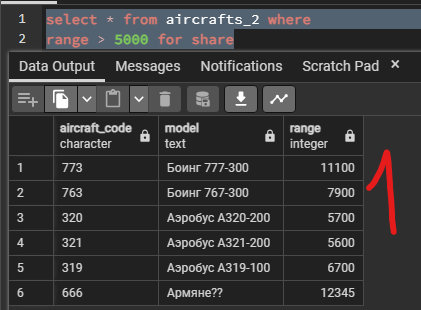




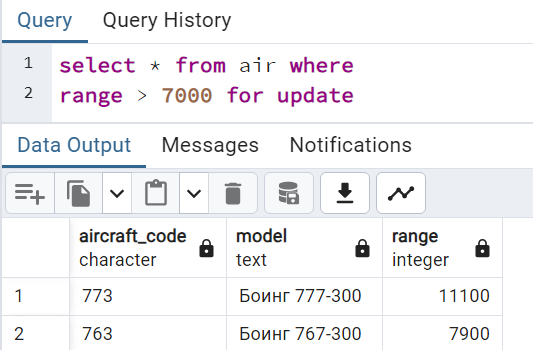
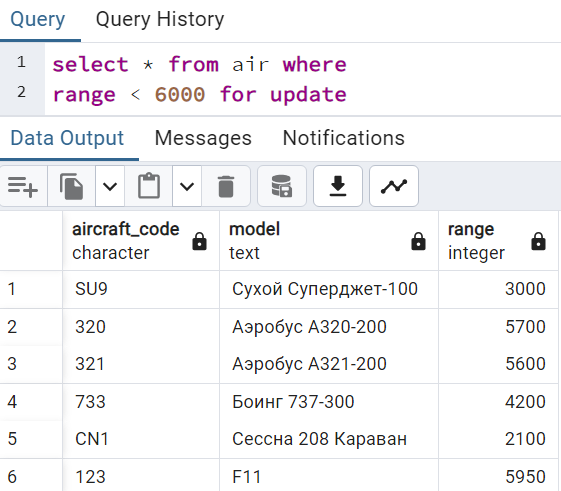
№5

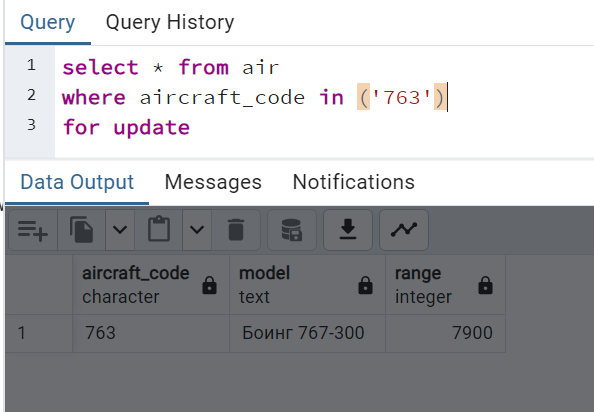
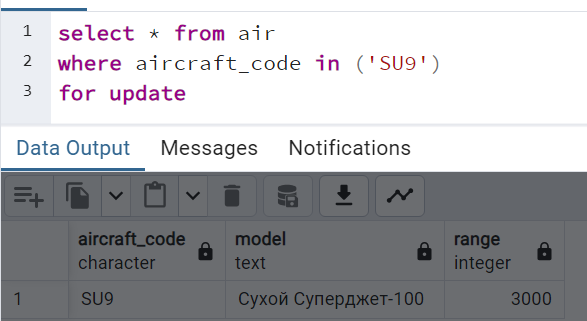
****

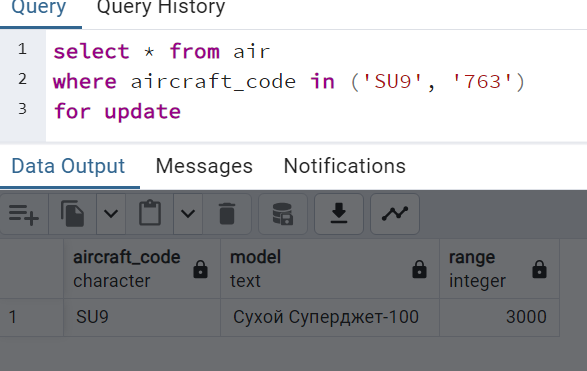
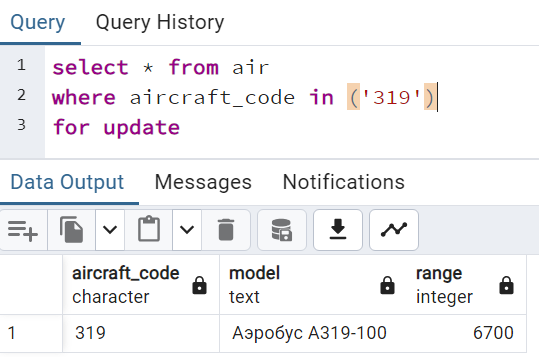
№6

****

№7

№8

Использование уровня изоляции Serializable гарантирует максимальную защиту данных от аномалий при одновременном доступе к ним несколькими транзакциями. Однако это может привести к увеличению времени блокировки и снижению параллелизма операций, особенно в ситуациях с большим числом одновременных транзакций бронирования. Вместо этого можно использовать явные блокировки для контроля доступа к критическим секциям или ресурсам, что позволит гибко управлять уровнем изоляции в зависимости от конкретных потребностей приложения и характеристик нагрузки. Этот подход более эффективен, поскольку позволяет избежать излишней конкуренции за блокировки и оптимизировать управление ресурсами. Однако он требует более внимательного управления транзакциями и блокировками, чтобы избежать возможных проблем, таких как взаимная блокировка и блокировка всего ресурса.

№9

В случае успешного примера, где обращение происходило к определенной строке данных, возможно применение сериализации. В первом случае, когда мы обращались к множествам HIGH и LOW, возникла ошибка. При последовательном выполнении транзакций мы получили бы аналогичный результат.

№10

1. При выполнении команды commit в первой транзакции, запросы обрабатываются последовательно. Точно так же происходит и с командой commit во второй транзакции. Однако может возникнуть ситуация, когда запросы первой транзакции еще не завершились, а запросы второй транзакции уже начали выполняться. Это нарушает принцип сериализации.

Использование индексации может предотвратить подобные ситуации, поскольку запросы двух транзакций не будут пересекаться. Кроме того, переход на уровень изоляции read committed может решить эту проблему, так как это позволит снизить ограничения для выполнения запросов.

1. Изменение значения из 13881 на 5572 в команде SELECT не вызовет проблем. Однако, если мы вернем значение обратно во второй транзакции, могут возникнуть проблемы. Ситуация может возникнуть, когда вторая транзакция выполнит SELECT раньше, чем первая транзакция сможет записать новую строку. Это происходит из-за уровня изоляции Serializable, который включает уровень изоляции Repeatable Read. Этот уровень блокирует данные, которые он читает, от изменений. Следовательно, первая транзакция не сможет записать новую строку до блокировки данных, где flight\_id = 13881. Использование индексации не поможет, так как она решает проблемы на уровне commit, а ошибка чтения возникает на уровне запроса первой транзакции.