

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc215780203)

[1 Описание предметной области](#_Toc215780204) 5

[2 Концептуальное проектирование БД 7](#_Toc215780205)

[3 Проектирование БД 11](#_Toc215780206)

[4 Реализация БД в СУБД *PostgreSQL* 18](#_Toc215780207)

[4.1 Создание таблиц и связей 18](#_Toc215780208)

[4.2 Создание индексов 21](#_Toc215780209)

[4.3 Заполнение таблиц данными 23](#_Toc215780210)

[4.4 Реализация ссылочной и семантической целостности данных 24](#_Toc215780211)

[4.5 Реализация необходимой точности значений расчётных данных 26](#_Toc215780212)

[5 Разработка запросов к БД 27](#_Toc215780213)

[5.1 Запросы на выборку данных 27](#_Toc215780214)

[5.2 Запросы с оконными функциями 37](#_Toc215780215)

[5.3 Запросы с вычисляемым полем 40](#_Toc215780216)

[5.4 Запросы с агрегатными функциями 42](#_Toc215780217)

[5.5 Запросы с подзапросами 44](#_Toc215780218)

[5.6 Запрос на основе материализованного представления 48](#_Toc215780219)

[6 Разработка объектов БД 50](#_Toc215780220)

[ЗАКЛЮЧЕНИЕ 61](#_Toc215780221)

# ВВЕДЕНИЕ

Современные компьютерные клубы используют значительные объемы средств вычислительной техники (СВТ), включающие игровые компьютеры, периферийные устройства, сетевое оборудование, игровые консоли и комплектующие. Эффективное управление такими ресурсами в условиях интенсивной эксплуатации требует структурированной системы хранения данных, позволяющей оперативно отслеживать перемещения оборудования между игровыми зонами, контролировать состояние техники, учитывать комплектующие для ремонта и апгрейдов, а также обеспечивать ответственность сотрудников за сохранность имущества. Реляционные базы данных на базе СУБД *PostgreSQL* обеспечивают надежность хранения, поддержку транзакций, механизмы проверки данных и возможность выполнения сложных аналитических запросов, что делает их подходящими для автоматизации учёта средств вычислительной техники в компьютерном клубе "КиберАрена".

Разработка базы данных *ComputerInventory* направлена на создание единой модели учёта СВТ компьютерного клуба, включающей категории оборудования, игровые ПК и консоли, комплектующие, инвентарные номера, перемещения между игровыми зонами и зоной ремонта, текущее состояние техники, а также работу с сотрудниками клуба (администраторами, техниками) и их должностями.

Цель работы – разработка реляционной базы данных *ComputerInventory* для автоматизации учёта средств вычислительной техники и операций с ними в компьютерном клубе "КиберАрена".

Для достижения цели были поставлены следующие задачи:

1. Провести анализ предметной области средств вычислительной техники компьютерного клуба и определить сущности, атрибуты и бизнес-правила.
2. Спроектировать реляционную БД в нотации *IDEF1X.*
3. Реализовать РБД в *PostgreSQL* с ограничениями целостности и заполнить тестовыми данными.
4. Разработать комплекс *SQL*-запросов и объекты реляционной БД (функции, процедуры, триггеры, курсоры) для эффективной работы системы.

# **1 Описание предметной области**

# Предметная область включает учёт всех видов средств вычислительной техники (СВТ) в компьютерном клубе "КиберАрена", их комплектующих, игровых зоны, ответственных сотрудников и истории перемещений оборудования. Для корректного функционирования системы необходимо фиксировать каждую операцию перемещения техники и поддерживать актуальное состояние инвентаря.

# Согласно анализу предметной области, к средствам вычислительной техники в компьютерном клубе относятся:

# Игровые компьютеры: высокопроизводительные ПК для одиночной и командной игры, оснащенные мощными видеокартами и процессорами.

# Игровые консоли: *PlayStation*, *Xbox* и *Nintendo Switch* для различных типов игр.

# Периферия: игровые мыши, клавиатуры, наушники, игровые кресла, мониторы с высокой частотой обновления.

# Сетевое оборудование: роутеры, коммутаторы и точки доступа для обеспечения стабильного интернет-соединения в игровом пространстве.

# Комплектующие для ремонта и апгрейда: видеокарты, процессоры, оперативная память, блоки питания, накопители.

# Серверное оборудование: игровые серверы для локальных сетевых игр.

# Основные сущности предметной области:

# Организация (филиал)

# Подразделение

# Сотрудник

# Категория СВТ

# Средство вычислительной техники

# Комплектующее

# Инвентарный номер

# Перемещение оборудование

# Текущее состояние

# Должность

# Состояние оборудования

# Тип перемещения

# В данной предметной области действуют следующие бизнес-правила:

# Одна организация содержит несколько подразделений.

# Каждое подразделение имеет ответственного сотрудника за учёт техники.

# Каждая единица СВТ имеет уникальный инвентарный номер.

# СВТ состоит из комплектующих.

# Перемещения оборудования фиксируются с указанием даты, подразделения и ответственного сотрудника.

# После каждого перемещения обновляется текущее местоположение техники.

# Все количества (в комплектующих) должны быть неотрицательными.

# История перемещений должна сохраняться полностью и не изменяться при корректировке текущего состояния.

# Должности сотрудников хранятся в отдельном справочнике для исключения дублирования данных.

# Сотрудник может занимать только одну должность из справочника должностей.

# Состояния оборудования хранятся в отдельном справочнике.

# Типы перемещений хранятся в отдельном справочнике.

# **2 Концептуальное проектирование БД**

На основе анализа предметной области выделены элементы данных. Проведён анализ синонимов и омонимов (синонимы и омонимы отсутствуют, все наименования уникальны).

Перечень атрибутов:

1. Идентификатор организации
2. Название организации
3. Адрес организации
4. Идентификатор подразделения
5. Название подразделения
6. Идентификатор сотрудника
7. ФИО сотрудника
8. Идентификатор должности
9. Контактная информация сотрудника
10. Идентификатор категории СВТ
11. Название категории СВТ
12. Описание категории СВТ
13. Идентификатор СВТ
14. Название СВТ
15. Модель СВТ
16. Год выпуска
17. Серийный номер
18. Идентификатор комплектующего
19. Название комплектующего
20. Модель комплектующего
21. Количество комплектующих
22. Является ли съемным
23. Идентификатор инвентарного номера
24. Номер инвентаря
25. Дата постановки на учёт
26. Способ приобретения
27. Идентификатор перемещения
28. Дата перемещения
29. Причина перемещения
30. Идентификатор типа перемещения
31. Идентификатор текущего состояния
32. Дата последнего обновления состояния
33. Идентификатор состояния оборудования
34. Идентификатор должности
35. Название должности
36. Описание должности
37. Диапазон заработной платы
38. Идентификатор состояния оборудования
39. Название состояния оборудования
40. Описание состояния оборудования
41. Идентификатор типа перемещения
42. Название типа перемещения
43. Описание типа перемещения

С использованием обратного метода построено множество функциональных зависимостей. Выполнена минимизация покрытия (удалены избыточные и транзитивные зависимости).

Минимальное покрытие функциональных зависимостей:

1 → 2, 3

4 → 1, 5

6 → 7, 8, 9

34 → 35, 36, 37

10 → 11, 12

13 → 10, 14, 15, 16, 17

18 → 13, 19, 20, 21, 22

23 → 13, 24, 25, 26

27 → 23, 4, 6, 28, 29, 30

31 → 23, 4, 32, 33

38 → 39, 40

41 → 42, 43

Объединение зависимостей и формирование отношений:

*R1.* Организациия*(Organization)*

1 → 2, 3

*R2.* Подразделение *(Department)*

4 → 1, 5

*R3.* Сотрудник *(Employee)*

6 → 7, 8, 9

*R4.* Должность *(Position)*

34 → 35, 36, 37

*R5.* Категория СВТ *(EquipmentCategorie)*

10 → 11, 12

*R6.* СВТ *(ComputingEquipment)*

13 → 10, 14, 15, 16, 17

*R7.* Комплектующее *(Component)*

18 → 13, 19, 20, 21, 22

*R8.* Инвентарный номер *(InventoryNumber)*

23 → 13, 24, 25, 26

*R9.* Перемещение *(Movement)*

27 → 23, 4, 6, 28, 29, 30

*R10.* Тип перемещенийя*(MovementType)*

41 → 42, 43

*R11.* Регистр ремонтных работ *(RegisterOfRepairWorks)*

31 → 23, 4, 32, 33

*R12.* Состояние оборудования *(EquipmentCondition)*

38 → 39, 40

Проверка выполнения 3НФ:

Первая нормальная форма (1НФ):

* все атрибуты атомарные (отсутствуют составные и многоуровневые атрибуты);
* нет повторяющихся групп (все данные представлены в виде отдельных записей без дублирования);
* определены первичные ключи для всех таблиц *(organization\_id, department\_id, position\_id, employee\_id, category\_id, equipment\_id, component\_id, inventory\_number\_id, movement\_id, repaip\_work\_id, condition\_id, movement\_type\_id).*

Вторая нормальная форма (2НФ):

* во всех таблицах неключевые атрибуты полностью зависят от первичных ключей;
* таблица *RegisterOfRepairWorks* имеет уникальный ключ по полю *inventory\_number\_id,* и все атрибуты полностью зависят от него;
* в таблице *Component* атрибуты названия, модели и количества полностью зависят от первичного ключа *component\_id;*
* в таблице *Movement* все атрибуты полностью зависят от первичного ключа *movement\_id;*
* в таблице *EquipmentCondition* все атрибуты полностью зависят от первичного ключа *condition\_id;*
* в таблице *MovementType* все атрибуты полностью зависят от первичного ключа *movement\_type\_id.*

Третья нормальная форма (3НФ):

* транзитивные зависимости вынесены в отдельные отношения (организации, подразделения, должности, категории СВТ, состояния оборудования, типы перемещений);
* поле *"position"* в таблице сотрудников докомпозировано в отдельную таблицу *Position* для устранения транзитивной зависимости;
* поле *"condition\_id"* в таблице текущего состояния докомпозировано в отдельную таблицу *EquipmentCondition* для устранения транзитивной зависимости;
* поле *"movement\_type\_id"* в таблице перемещений докомпозировано в отдельную таблицу *MovementType* для устранения транзитивной зависимости;
* таблицы *Component* и *InventoryNumber* содержат только атрибуты, непосредственно относящиеся к комплектующим и инвентарным номерам соответственно;
* таблицы *Movement* и *RegistrOfRepairWorks* содержат только атрибуты, непосредственно относящиеся к операциям перемещения и отслеживания состояния.

База данных удовлетворяет 3НФ, зависимости сохранены.

# **3 Проектирование БД**

Логическая модель данных в нотации *IDEF1X* спроектирована в draw.io и представлена на рисунке 1. Выделены сущности: Организация, Подразделение, Сотрудник, Должность, Категория СВТ, СВТ, Комплектующее, ИнвентарныйНомер, Перемещение, РегистРемонтныхРабот. Между ними установлены неидентифицирующие связи мощностью «один-ко-многим».

Логическая модель данных, выполненная в среде разработки *draw.io* в нотации *IDEF1X* для СУБД PostgreSQL представлена на рисунке 1.

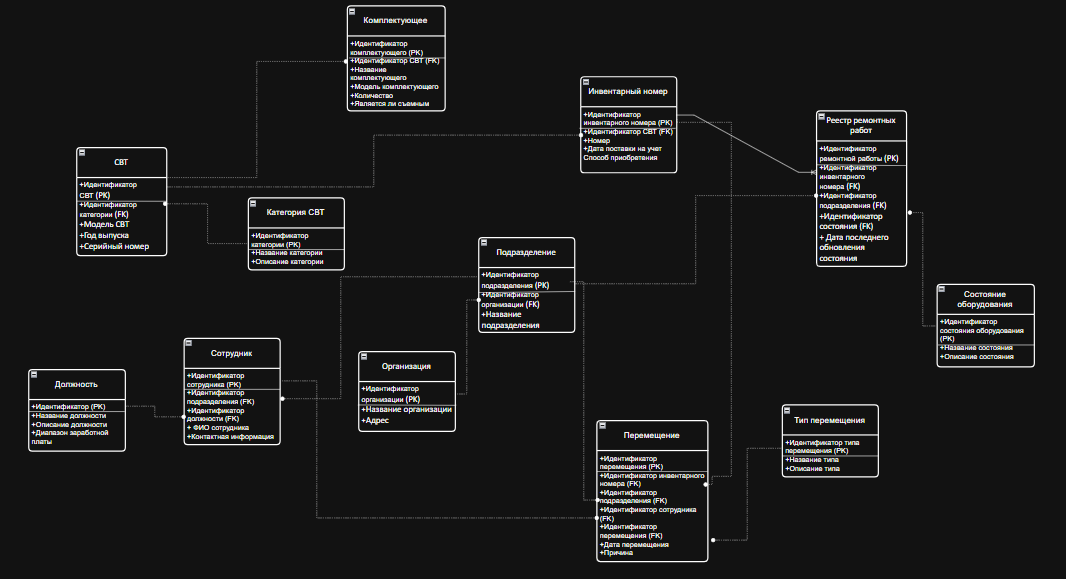


Рисунок 1 – Логическая модель данных

Физическая модель данных, выполненная в среде разработки *draw.io* в нотации *IDEF1X* для СУБД *PostgreSQL* представлена на рисунке 2.

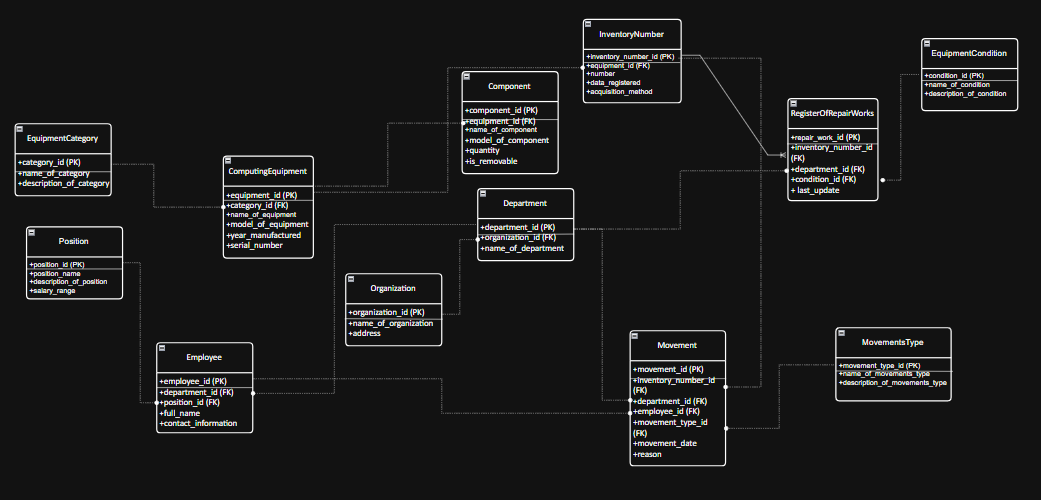


Рисунок 2 – Физическая модель данных

# База данных *ComputerInventory* состоит из двенадцати таблиц:

# *Organization* – справочник организаций.

# *Department* – подразделения.

# *Position* – должности сотрудников.

# *Employee* – сотрудники.

# *EquipmentCategorie* – категории средств вычислительной техники.

# *ComputingEquipment* – все виды средств вычислительной техники.

# *Component* – комплектующие.

# *InventoryNumber* – инвентарные номера.

# *MovementType* – типы перемещений оборудования.

# *Movement* – перемещения оборудования.

# *EquipmentCondition* – состояния оборудования.

# *RegisrOfRepairWorks* – регистр ремонтных работ.

# Описание таблиц базы данных *ComputerInventory:*

# Таблица *Organization:*

# *organization\_id* – уникальный идентификатор организации, *integer, not null, primary key*

# *name\_of\_organization* – название организации, *text, not null*

# *address* – адрес организации, *text, not null*

# Таблица *Department:*

# *department\_id* – уникальный идентификатор подразделения, *integer, not null, primary key*

# *organization\_id* – идентификатор организации, *integer, not null, foreign key references Organization(organization\_id) on update cascade on delete cascade*

# *name\_of\_department* – название подразделения, *text, not null*

# Таблица *Position:*

# *position\_id* – уникальный идентификатор должности, *integer, not null, primary key*

# *name\_of\_position* – название должности, *text, not null, unique*

# *description\_of\_position* – описание должности, *text*

# *salary\_range* – диапазон заработной платы, *text*

# Таблица *Employee:*

# *employee\_id* – уникальный идентификатор сотрудника, *integer, not null, primary key*

# *department\_id* – идентификатор подразделения, *integer, not null, foreign key references Department(department\_id) on update cascade on delete cascade*

# *position\_id* – идентификатор должности, *integer, not null, foreign key references Position(position\_id) on update cascade on delete cascade*

# *full\_name* – ФИО сотрудника, *text, not null*

# *contact\_information* – контактная информация, *text*

# Таблица *EquipmentCategory:*

# *category\_id* – уникальный идентификатор категории СВТ, *integer, not null, primary key*

# *name\_of\_category* – название категории, *text, not null*

# *description\_of\_category* – описание категории, *text*

# Таблица *ComputingEquipment:*

# *equipment\_id* – уникальный идентификатор СВТ, *integer, not null, primary key*

# *category\_id* – идентификатор категории, *integer, not null, foreign key references EquipmentCategorie(category\_id) on update cascade on delete cascade*

# *name\_of\_category* – название СВТ, *text, not null*

# *model\_of\_category* – модель СВТ, *text, not null*

# *year\_manufactured* – год выпуска, *integer, not null, check (year\_manufactured* > 1980)

# *serial\_number* – серийный номер, *text, not null*

# Таблица *Component:*

# *component\_id* – уникальный идентификатор комплектующего, *integer, not null, primary key*

# *equipment\_id* – идентификатор СВТ, *integer, not null, foreign key references ComputingEquipment(equipment\_id) on update cascade on delete cascade*

# *name\_of\_component* – название комплектующего, *text, not null*

# *model\_of\_component* – модель комплектующего, *text, not null*

# *quantity* – количество, *integer, not null, check (quantity* >= 0)

# *is\_removable* – является ли съемным, *boolean, not null*

# Таблица *InventoryNumber:*

# *inventory\_number\_id* – уникальный идентификатор инвентарного номера, *integer, not null, primary key*

# *equipment\_id* – идентификатор СВТ, *integer, not null, foreign key references ComputingEquipment(equipment\_id) on update cascade on delete cascade*

# *number* – инвентарный номер, *text, not null, unique*

# *date\_registered* – дата постановки на учет, *date, not null*

# *acquisition\_method* – способ приобретения, *text*

# Таблица *MovementType:*

# *movement\_type\_id* – уникальный идентификатор типа перемещения, *integer, not null, primary key*

# *name\_of\_type* – название типа перемещения, *text, not null, unique*

# *description\_of\_type* – описание типа перемещения, *text*

# Таблица *Movement:*

# *movement\_id* – уникальный идентификатор перемещения, *integer, not null, primary key*

# *inventory\_number\_id* – идентификатор инвентарного номера, *integer, not null, foreign key references InventoryNumber(inventory\_number\_id) on update cascade on delete cascade*

# *department\_id* – идентификатор подразделения, *integer, not null, foreign key references Department(department\_id) on update cascade on delete cascade*

# *employee\_id* – идентификатор сотрудника, *integer, not null, foreign key references Employee(employee\_id) on update cascade on delete cascade*

# *movement\_type\_id* – идентификатор типа перемещения, *integer, not null, foreign key references MovementType(movement\_type\_id) on update cascade on delete cascade*

# *movement\_date* – дата перемещения, *date, not null*

# *reason* – причина перемещения, *text*

# Таблица *EquipmentCondition:*

# *condition\_id* – уникальный идентификатор состояния оборудования, *integer, not null, primary key*

# *name\_of\_condition* – название состояния, *text, not null, unique*

# *description\_of\_condition* – описание состояния, *text*

# Таблица *RegisrOfRepairWorks*:

# *repair\_work\_id* – уникальный идентификатор состояния, *integer, not null, primary key*

# *inventory\_number\_id* – идентификатор инвентарного номера, *integer, not null, foreign key references InventoryNumber(inventory\_number\_id) on update cascade on delete cascade, unique*

# *department\_id* – идентификатор подразделения, *integer, not null, foreign key references Department(deppartment\_id) on update cascade on delete cascade*

# *condition\_id* – идентификатор состояния, *integer, not null, foreign key references EquipmentCondition(condition\_id) on update cascade on delete cascade*

# *last\_update* – дата последнего обновления состояния, *date, not null*

# **4 Реализация БД в СУБД *PostgreSQL***

# **4.1 Создание таблиц и связей**

База данных *ComputerInventory* реализована в СУБД *PostgreSQL* 17, ее создание можно увидеть на рисунке 3.

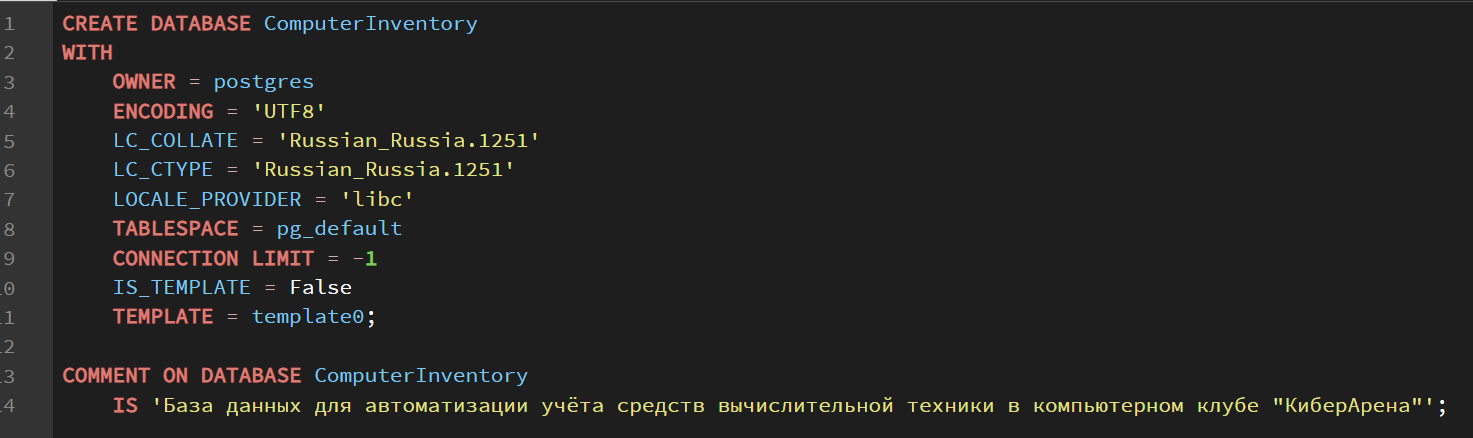


Рисунок 3 – Создание базы данных

Все таблицы созданы с помощью *DDL*-команд, полностью соответствуют физической модели и содержат необходимые ограничения целостности.

Основные особенности реализации:

⎯ Все таблицы имеют первичные ключи (*PK*), реализованные через *SERIAL*-типы;

⎯ Ссылочная целостность обеспечивается внешними ключами (*FK*) с правильным указанием связей между таблицами;

⎯ Для всех внешних ключей установлены правила *on update cascade on delete cascade* – при удалении организации автоматически удаляются все подразделения, при удалении подразделения удаляются связанные сотрудники и записи в регистрах ремонтных работ, при удалении СВТ удаляются комплектующие и инвентарные номера;

⎯ Ограничение *check* (*quantity* >= 0) в таблице *Component* запрещает отрицательные значения количества комплектующих;

⎯ Ограничение *check* (*year\_manufactured* > 1980) в таблице *ComputingEquipment* гарантирует корректные значения года выпуска оборудования;

⎯ Уникальные ограничения (*unique*) для полей *name\_of\_position,* *number* (инвентарный номер), *name\_of\_condition* и других гарантируют отсутствие дублирования данных;

⎯ Таблица *RegisterOfRepairWorks* использует уникальный внешний ключ (*inventory\_number\_id*), что обеспечивает однозначное соответствие между инвентарным номером и записью о состоянии оборудования.

*SQL*-скрипты создания таблиц, представленные на рисунках 4, 5 ,6 и 7 были успешно выполнены в *pgAdmin* 4.

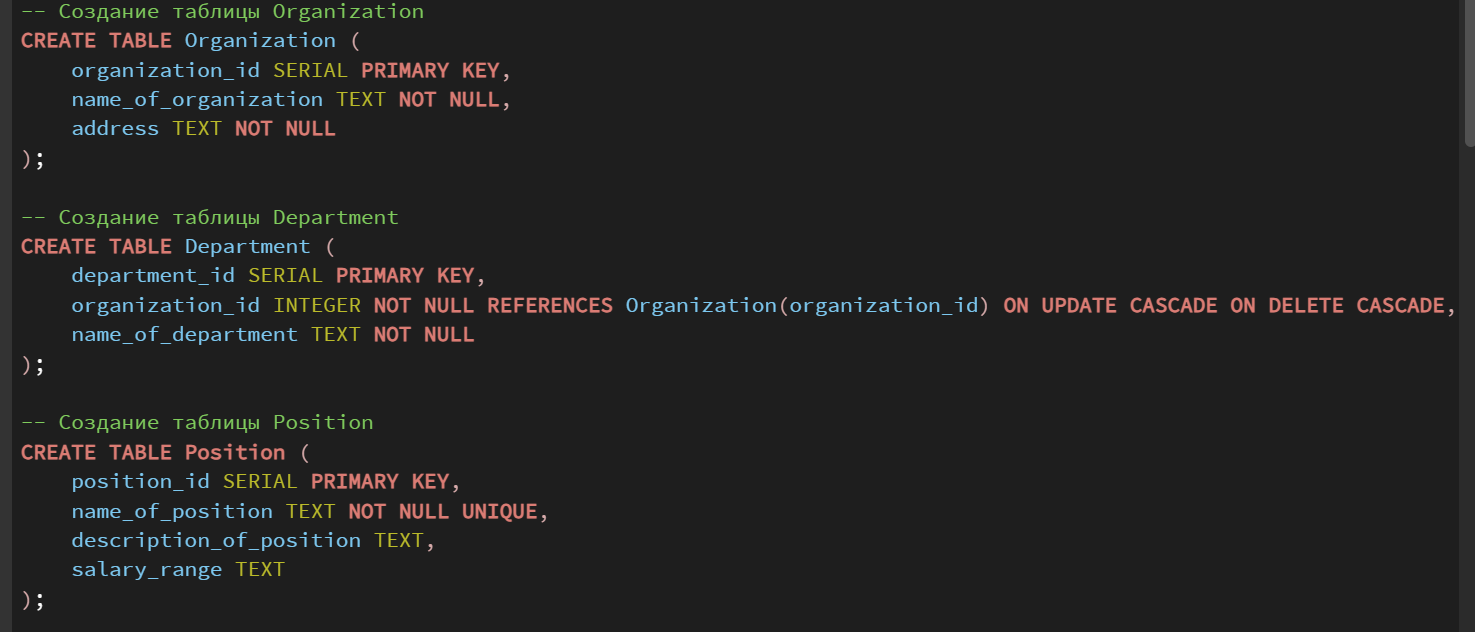


Рисунок 4 – Создание таблиц

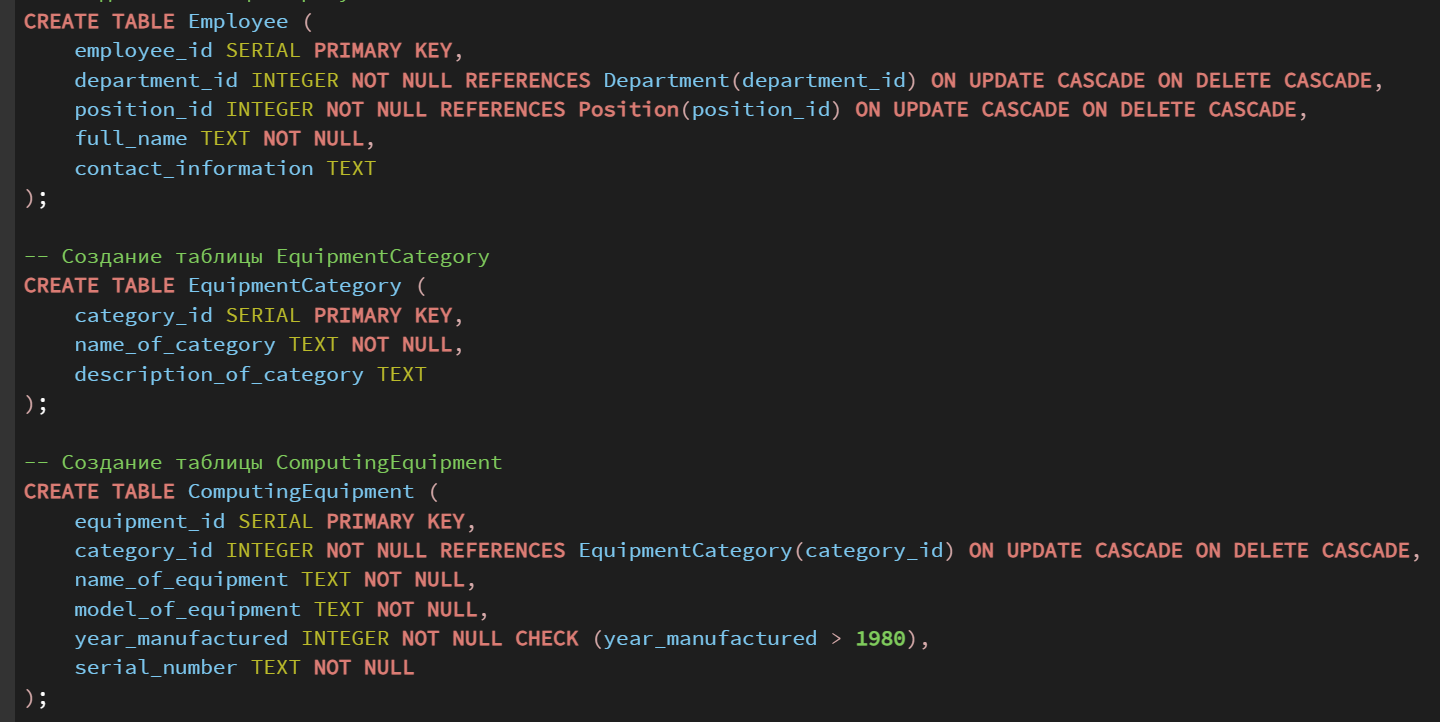


Рисунок 5 – Создание таблиц

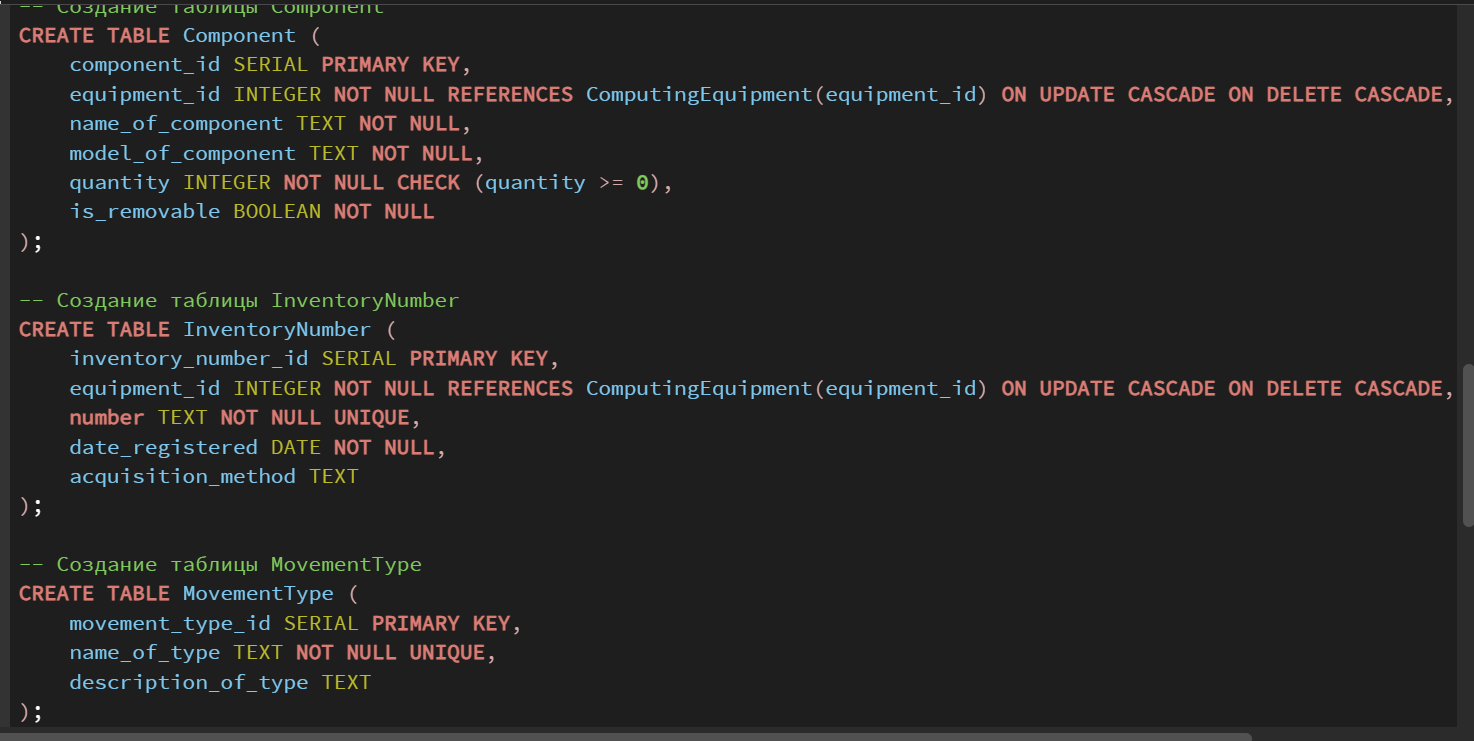


Рисунок 6 – Создание таблиц

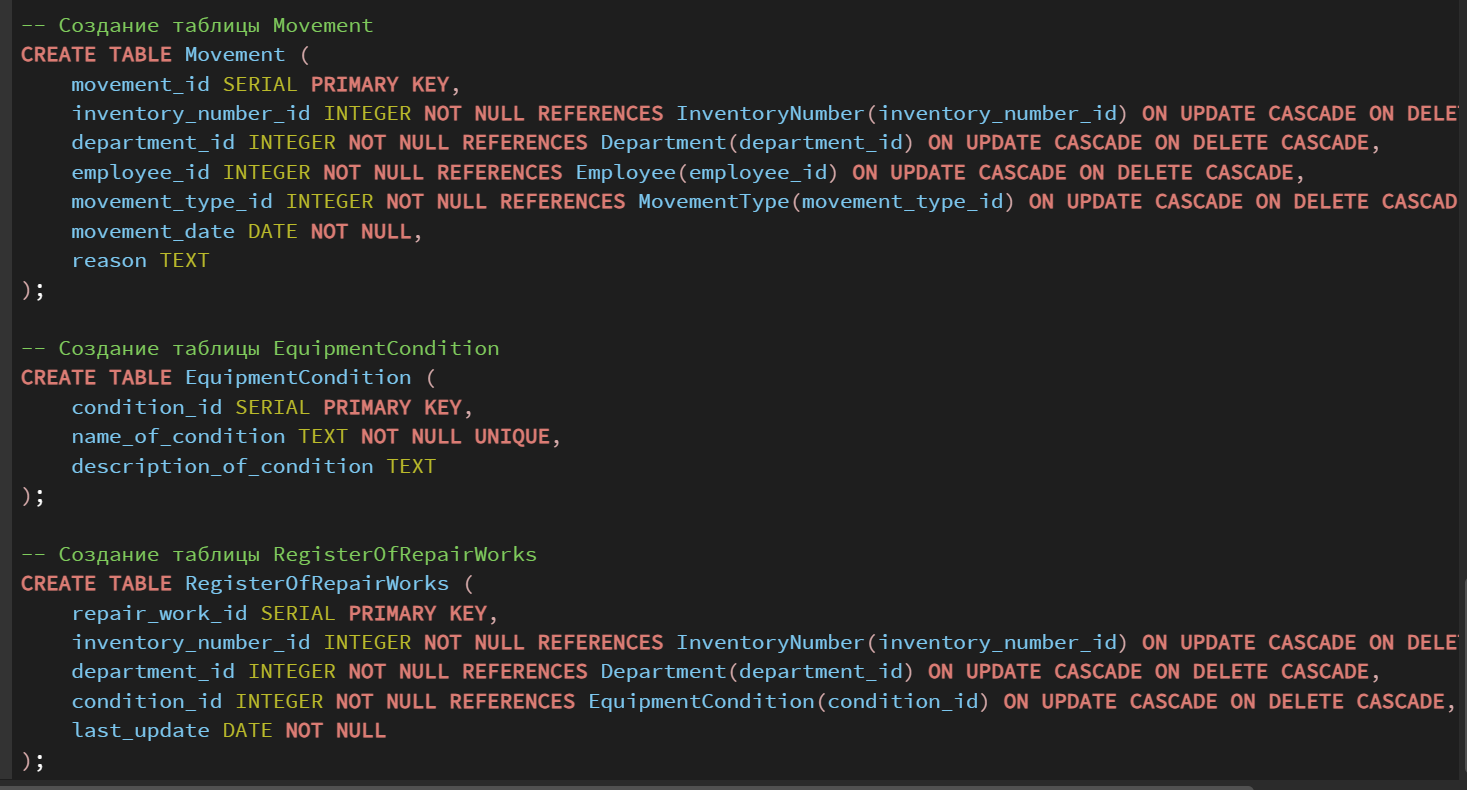


Рисунок 7 – Создание таблиц

# **4.2 Создание индексов**

Для повышения производительности запросов в базе данных, предназначенной для управления нефтебазой, были созданы индексы. Первичные ключи всех таблиц индексируются автоматически *PostgreSQL*, однако для оптимизации операций выборки, фильтрации и соединений дополнительно созданы индексы по внешним ключам и полям, наиболее часто участвующим в аналитических запросах.

Дополнительно были созданы индексы для внешних ключей и часто используемых полей:

*⎯ ComputingEquipment(category\_id) –* для быстрого поиска оборудования по категориям;

*⎯ Component(equipment\_id) –* для ускорения запросов к комплектующим конкретного оборудования;

*⎯ InventoryNumber(equipment\_id) –* для оперативного получения инвентарных номеров по оборудованию;

*⎯ Movement(inventory\_number\_id) –* для анализа истории перемещений конкретного инвентарного номера;

*⎯ Movement(department\_id) –* для получения всех перемещений оборудования в определенном подразделении;

*⎯ Movement(movement\_date) –* для фильтрации перемещений по временным периодам;

*⎯ Movement(employee\_id) –* для отслеживания действий конкретного сотрудника;

*⎯ RegisterOfRepairWorks(condition\_id) –* для анализа оборудования по состоянию;

*⎯ Employee(department\_id) –* для получения списка сотрудников конкретного подразделения.

Создание этих индексов можно увидеть на рисунке 8.

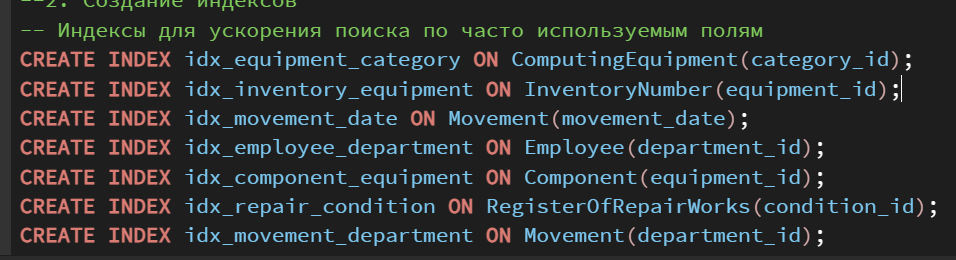


Рисунок 8 – Создание индексов

# **4.3 Заполнение таблиц данными**

Для проверки корректности структуры, ограничений целостности и связей между таблицами база данных *ComputerInventory* была заполнена демонстрационными данными.

Вставка данных выполнялась с помощью операторов *INSERT INTO* языка *DML*.

Данные добавлены во все двенадцать таблиц в правильной последовательности с учётом внешних ключей:

⎯ организации и подразделения компьютерного клуба "КиберАрена", ⎯ категории оборудования и сами средства вычислительной техники (игровые ПК, консоли, периферия),

⎯ комплектующие, инвентарные номера и справочники (типы перемещений, состояния оборудования, должности),

⎯ сотрудники клуба и их контактная информация,

⎯ операции перемещения оборудования между зонами и записи в регистре ремонтных работ.

Все ограничения *NOT NULL, CHECK* (*quantity* >= 0), *CHECK* (*year\_manufactured* > 1980) и внешние ключи успешно прошли проверку – ошибок ссылочной целостности не возникло. При попытке вставить некорректные данные (отрицательное количество комплектующих, год выпуска оборудования до 1980 года, несуществующие идентификаторы внешних ключей) СУБД корректно отклоняла операции с соответствующими сообщениями об ошибках.

После заполнения выполнены контрольные запросы *SELECT \* FROM* для каждой таблицы для визуальной проверки корректности добавленных данных.

Пример содержимого таблицы *ComputingEquipment* после заполнения представлен на рисунке 9.



Рисунок 9 – Содержимое таблицы ComputingEquipment после заполнения

# **4.4 Реализация ссылочной и семантической целостности данных**

В разработанной базе данных нефтебазы механизмы ссылочной и семантической целостности реализованы непосредственно в процессе создания таблиц посредством средств языка *DDL* *PostgreSQL*. Все связи между сущностями, а также ограничения корректности данных были заложены в структуру таблиц и автоматически контролируются СУБД.

Ссылочная целостность обеспечивается использованием внешних ключей (*foreign key*), связывающих таблицы между собой и определяющих допустимые операции над связанными записями. Для всех *FK* заданы правила:

* *on update cascade* – автоматическое обновление значения внешнего ключа при изменении ключа в родительской таблице;
* *on delete cascade* – автоматическое удаление зависимых записей при удалении родительской записи.

Таким образом, изменение или удаление записи в таблицах *oilbase*, *warehouse*, *oilproduct*, *supplier*, *client* автоматически отражается на таблицах *supply*, *shipment* и *stockbalance*, что гарантирует сохранение целостности.

Связи были определены непосредственно в *DDL* при создании каждой таблицы, например:

* таблица *warehouse* ссылается на *oilbase*;
* *supply* содержит ссылочные ключи на *oilproduct*, *warehouse* и *supplier*;
* shipment содержит *FK* на *oilproduct*, *warehouse* и *client*;
* *stockbalance* использует составной первичный ключ и одновременно *FK* на *warehouse* и *oilproduct*.

Такой подход обеспечивает строгую структурную согласованность данных и предотвращает нарушение целостности между зависимыми объектами.

Семантическая целостность данных реализована с помощью следующих ограничений:

* *not* *null* – применяется ко всем обязательным атрибутам (название нефтепродукта, склад, даты, количество);
* *check* (*quantity* >= 0) – предотвращает появление отрицательных значений в количественных полях таблиц *supply*, *shipment* и *stockbalance*;
* *primary* *key* – обеспечивает уникальность ключевых полей;
* *unique* (*warehouse\_id, oilproduct\_id*) в *stockbalance* реализовано в составе составного первичного ключа и исключает дублирование остатков по одному товару на одном складе.

Данные механизмы обеспечивают корректность вводимых значений и предотвращают появление логически неверных данных.

# **4.5 Реализация необходимой точности значений расчётных данных**

# Для обеспечения корректности операций учёта средств вычислительной техники и исключения ошибок при вычислениях в базе данных компьютерного клуба предусмотрена повышенная точность хранения всех числовых значений. В *PostgreSQL* для таких полей использован тип *numeric(precision, scale),* позволяющий хранить значения с фиксированным числом знаков после запятой и обеспечивающий точное арифметическое вычисление, что особенно важно при работе с комплектующими, стоимостью ремонта и формированием финансовой отчётности.

# В разработанной структуре базы данных применены следующие форматы числовых полей:

# ***integer*** – для количественных показателей комплектующих в таблице *Component (quantity),* с дополнительным ограничением *CHECK* (*quantity* >= 0), гарантирующим недопустимость отрицательных значений. Этот тип обеспечивает точный учёт количества деталей для ремонта и апгрейдов оборудования. Для года выпуска техники (*year\_manufactured)* также используется *integer* с ограничением *CHECK* (*year\_manufactured* > 1980), что исключает некорректные значения и обеспечивает достоверность данных об оборудовании.

# ***numeric(12,2)*** – для денежных значений, связанных с расчётами стоимости ремонта, апгрейдов и закупок оборудования. Хотя в текущей версии схемы это поле не представлено явно в таблицах, оно используется в разработанных функциях (например, в функции *calculate\_repair\_cost*) для расчёта стоимости ремонтных работ с точностью до копеек, что соответствует финансовым стандартам и требованиям бухгалтерского учёта.

# Эти типы данных гарантируют корректность при расчётах итоговых сумм, отчётов и статистических показателей, а также обеспечивают совместимость с финансовыми стандартами и требованиями внутреннего контроля компьютерного клуба "КиберАрена".

# **5 Разработка запросов к БД**

# **5.1 Запросы на выборку данных**

Однотабличный запрос с простым условием. Создадим запрос для получения списка сотрудников, занимающих должность администратора. Запрос извлекает идентификатор сотрудника, ФИО и контактную информацию из таблицы *Employee,* применяя простое условие отбора: выбираются только те сотрудники, у которых *position\_id* соответствует идентификатору должности "Администратор". Данный запрос позволяет быстро идентифицировать ответственных лиц за управление клиентами и бронированием в компьютерном клубе. Запрос и его результат выполнения представлен на рисунке 10.

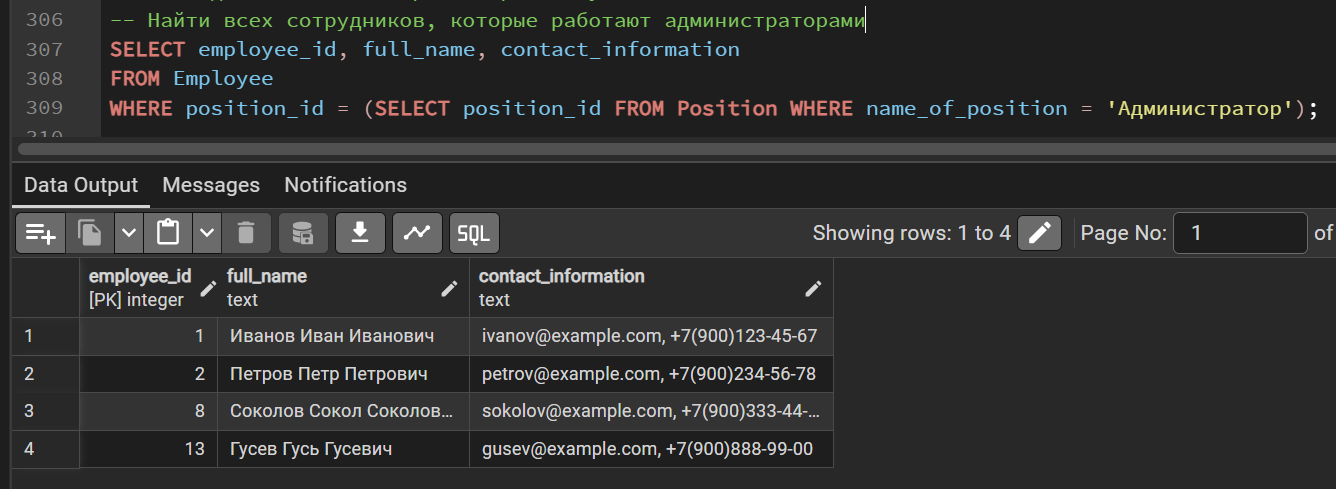


Рисунок 10 – Однотабличный запрос с простым условием

Однотабличный запрос со сложным условием *and, and, not.* Создадим запрос для поиска современных игровых компьютеров, исключая бюджетные модели. Запрос выбирает записи из таблицы *ComputingEquipment,* где год выпуска оборудования больше 2020 года (*year\_manufactured* > 2020), категория соответствует игровым компьютерам (*category\_id IN* подзапрос) и одновременно исключает оборудование с пометкой *"Digital"* в серийном номере (*serial\_number NOT LIKE '%Digital%'*). Данный запрос позволяет администратору клуба оперативно получать информацию о топовых игровых ПК, не включая в выборку урезанные версии консолей. Запрос и его результат выполнения представлен на рисунке 11.

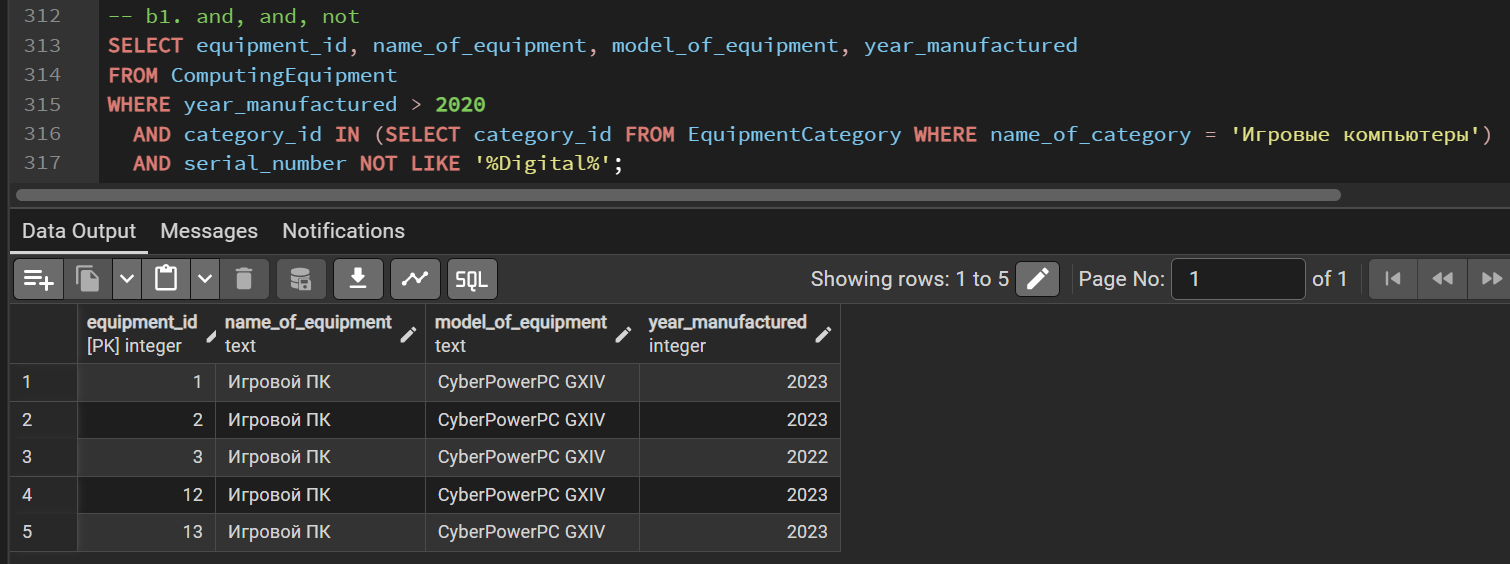


Рисунок 11 – Однотабличный запрос, со сложным условием and, and, not

Однотабличный запрос со сложным условием *and, or*. Создадим запрос для получения списка инвентарных номеров, зарегистрированных в определенные периоды. Запрос выбирает записи из таблицы *InventoryNumber*, где либо способ приобретения - "Покупка" и дата регистрации после 1 января 2023 года, либо дата регистрации находится в декабре 2022 года. Данный запрос позволяет отслеживать оборудование, приобретенное в разные периоды, что помогает в формировании отчетов о закупках за конкретные временные интервалы. Запрос и его результат выполнения представлен на рисунке 12.

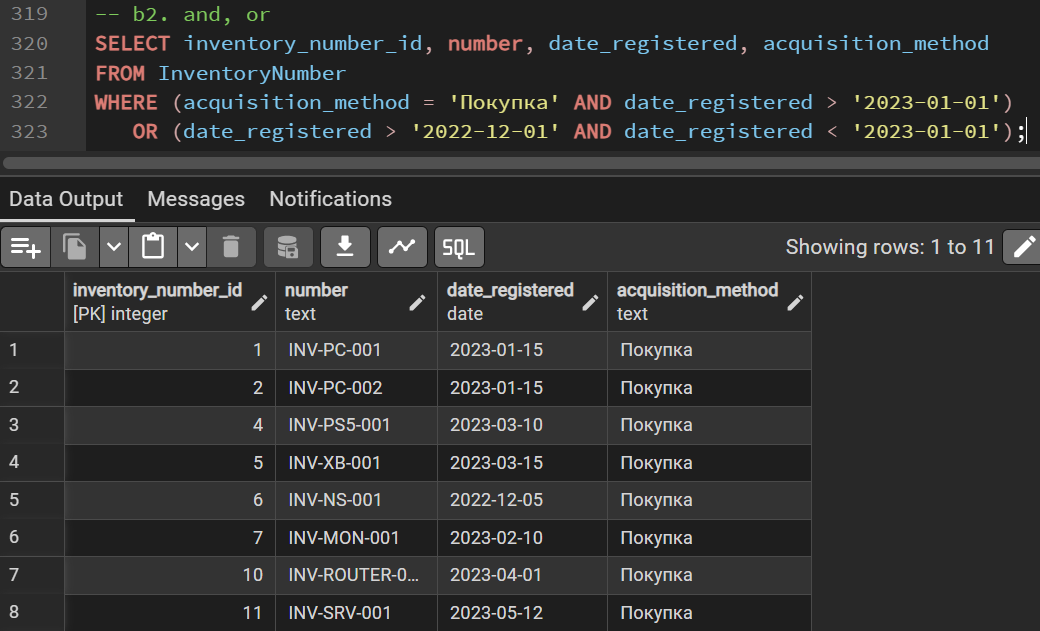


Рисунок 12 – Однотабличный запрос, со сложным условием and, or

Однотабличный запрос со сложным условием *and, and.* Создадим запрос для анализа ремонтных работ оборудования в первом полугодии 2023 года. Запрос извлекает записи из таблицы *Movement,* где дата перемещения находится в диапазоне с 1 января по 30 июня 2023 года (*movement\_date BETWEEN* '2023-01-01' *AND* '2023-06-30') и тип операции соответствует ремонту (*movement\_type\_id* = 3). Данный запрос позволяет сервисному центру отслеживать все ремонтные работы, выполненные в первом полугодии, для формирования отчетов по обслуживанию оборудования и планирования бюджета на следующий период. Запрос и его результат выполнения представлен на рисунке 13.

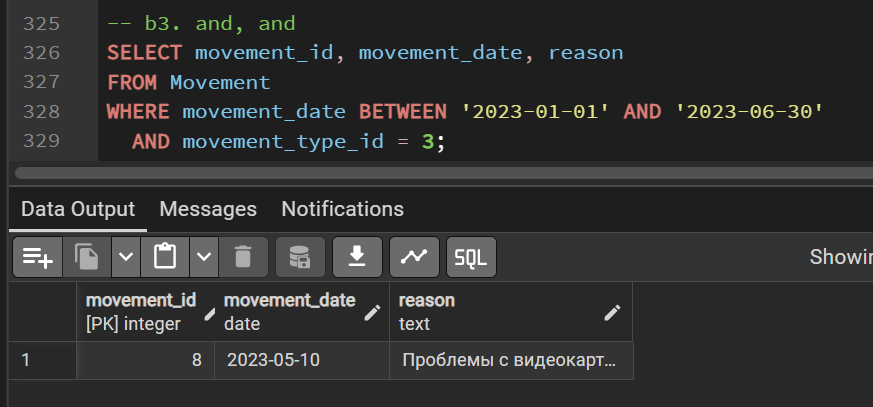


Рисунок 13 – Однотабличный запрос, со сложным условием and, and

Многотабличный запрос с вычисляемым полем и простым условием. Создадим запрос для получения списка современного игрового оборудования с расчетом возраста в годах. Запрос объединяет таблицы *ComputingEquipment* и *EquipmentCategory*, вычисляет возраст оборудования как разницу между текущим годом и годом выпуска (*EXTRACT(YEAR FROM CURRENT\_DATE*) - *ce.year\_manufactured AS age\_years),* и фильтрует результаты, оставляя только оборудование возрастом не более 4 лет. Данный запрос позволяет администратору компьютерного клуба оперативно получать информацию о новом оборудовании для планирования обновлений парка техники и формирования отчетов о возрастном составе средств вычислительной техники. Запрос и его результат выполнения представлен на рисунке 14.

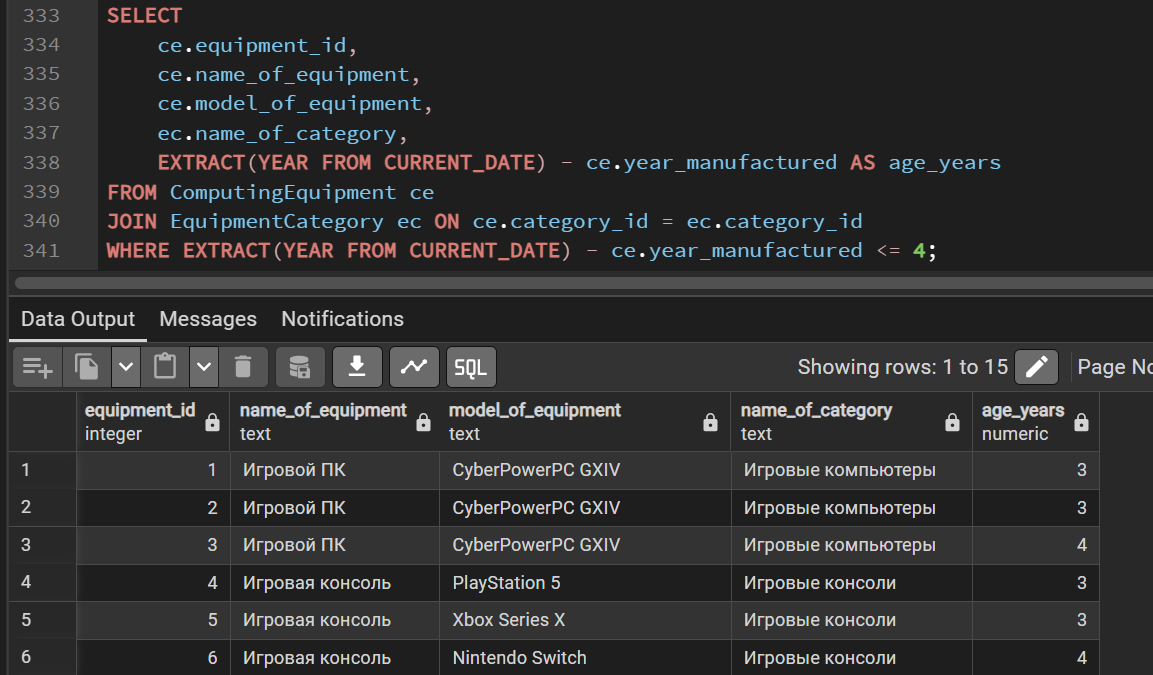


Рисунок 14 – Многотабличный запрос с вычисляемым полем и простым условием

Многотабличный запрос со сложным условием *or, and, not*. Создадим запрос для анализа операций обслуживания оборудования в компьютерном клубе. Запрос объединяет таблицы *Movement*, *MovementType* и *Employee*, выбирая перемещения оборудования, где тип операции - либо "Ремонт", либо "Апгрейд" (*mt.name\_of\_type* = 'Ремонт' *OR mt.name\_of\_type* = 'Апгрейд'), дата операции после 1 января 2023 года (*m.movement\_date* > '2023-01-01') и при этом исключает работы, выполненные сотрудниками на должности "Уборщик" (*e.position\_id* != подзапрос). Данный запрос позволяет руководству сервисного центра отслеживать все технические работы с оборудованием, выполняемые квалифицированным персоналом, и анализировать загруженность специалистов в указанный период. Запрос и его результат выполнения представлен на рисунке 15.

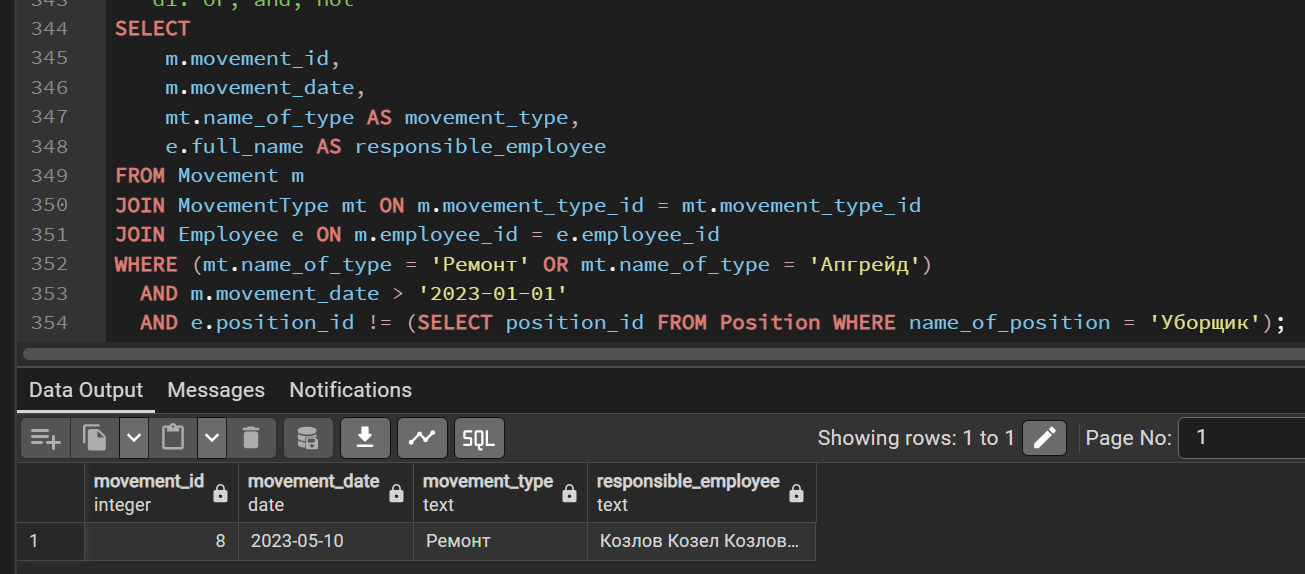


Рисунок 15 – Многотабличный запрос, со сложным условием or, and, not

Многотабличный запрос со сложным условием *and, and.* Создадим запрос для получения информации о текущем расположении современного оборудования 2023 года выпуска. Запрос объединяет таблицы *ComputingEquipment, InventoryNumber, Movement* и *Department,* выбирая записи, где дата перемещения является последней для данного инвентарного номера (*m.movement\_date* = подзапрос с *MAX*) и год выпуска оборудования равен 2023 (*ce.year\_manufactured* = 2023). Данный запрос позволяет администратору клуба оперативно получать актуальную информацию о расположении самого нового оборудования в различных зонах компьютерного клуба, что помогает в оптимизации распределения техники и планировании турниров. Запрос и его результат выполнения представлен на рисунке 16.

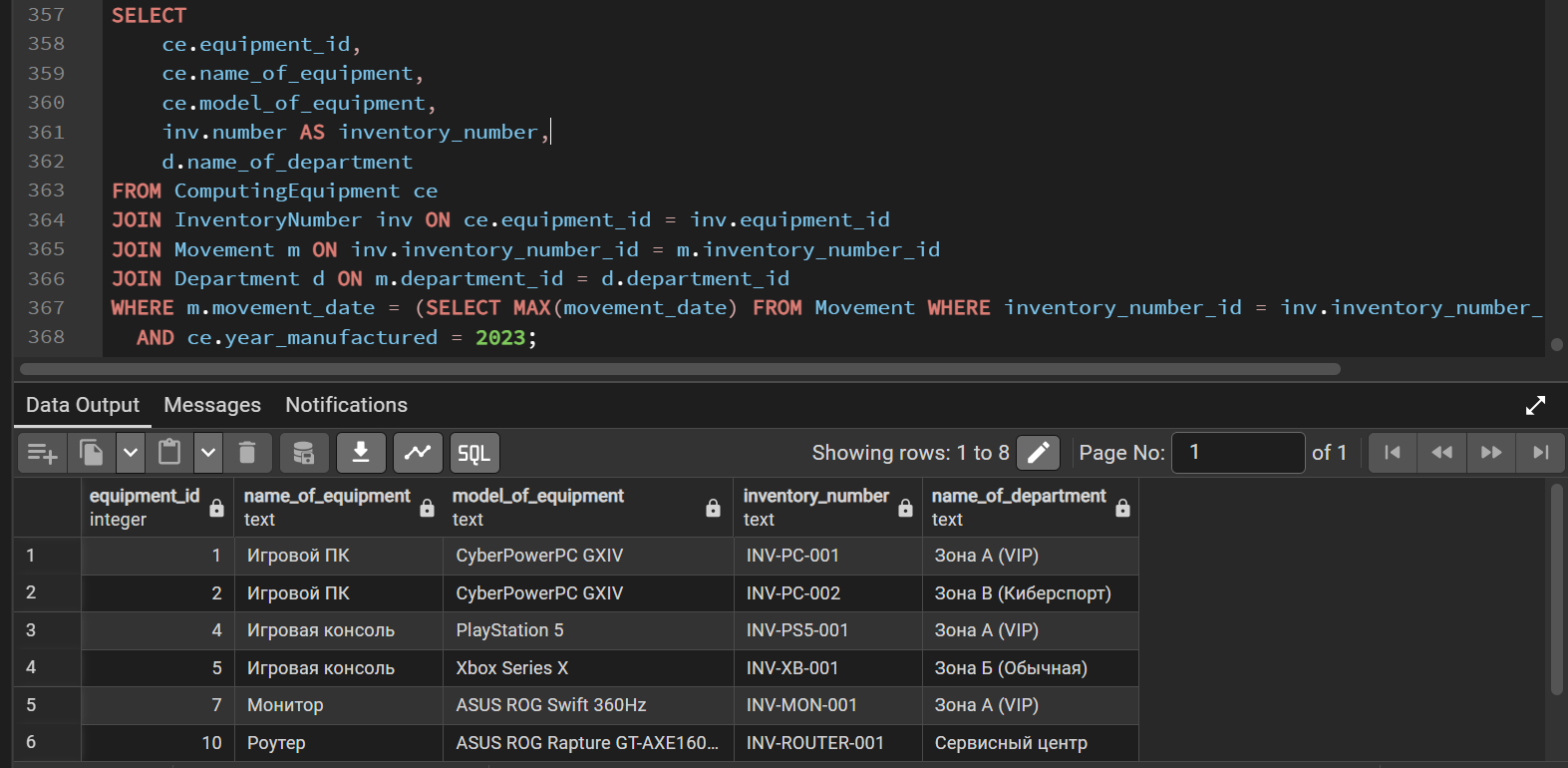


Рисунок 16 – Многотабличный запрос, со сложным условием and, and

Многотабличный запрос со сложным условием *and* и оператором *between.* Создадим запрос для получения информации о перемещениях оборудования в административном отделе за конкретный период. Запрос объединяет таблицы *Movement, InventoryNumber, ComputingEquipment, MovementType* и *Department*, выбирая записи, где дата перемещения находится в диапазоне с 10 по 12 ноября 2025 года (*m.movement\_date* *BETWEEN* ' 2023-01-01' *AND* ' 2023-03-31') и оборудование перемещалось в административное подразделение (*m.department\_id* = 1). Данный запрос позволяет руководству компьютерного клуба анализировать активность по перемещению техники в административной зоне за короткий временной период, что помогает в контроле за распределением оборудования и оптимизации рабочих процессов. Запрос и его результат выполнения представлен на рисунке 17.

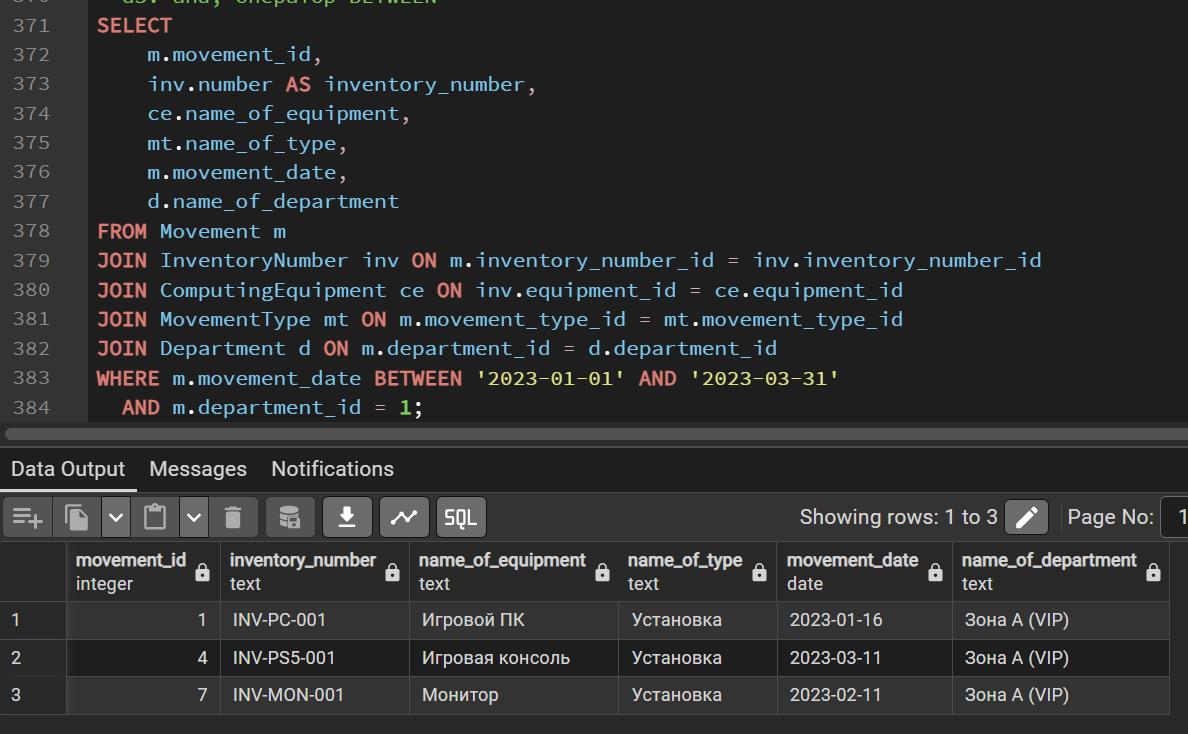


Рисунок 17 – Многотабличный запрос, со сложным условием and и оператором between

Многотабличный запрос со сложным условием *and* и оператором *not between.* Создадим запрос для анализа игровых консолей, не входящих в основной период закупок. Запрос объединяет таблицы *ComputingEquipment* и *EquipmentCategory,* выбирая оборудование, год выпуска которого не попадает в период 2020-2022 годов (*ce.year\_manufactured NOT BETWEEN* 2020 *AND* 2022) и относящееся к категории "Игровые консоли" (*ec.name\_of\_category* = 'Игровые консоли'). Данный запрос позволяет руководству компьютерного клуба идентифицировать редкие или устаревшие модели консолей, которые не были приобретены в основной период обновления парка, что помогает в планировании специализированных закупок для коллекционных или ретро-игр. Запрос и его результат выполнения представлен на рисунке 18.

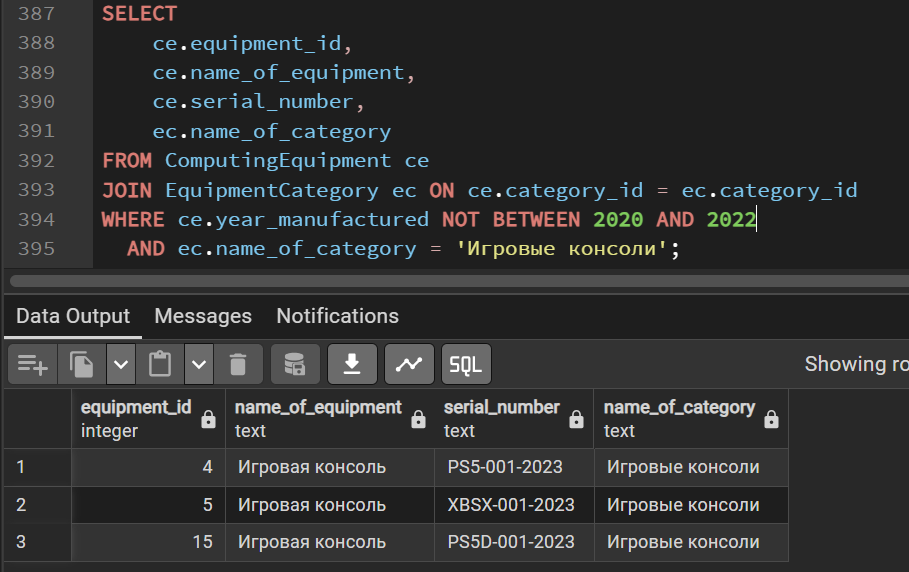


Рисунок 18 – Многотабличный запрос, со сложным условием and и оператором not between

Многотабличный запрос с оператором *in.* Создадим запрос для получения информации о ключевых сотрудниках компьютерного клуба. Запрос объединяет таблицы *Employee, Position* и *Department*, выбирая записи, где должность сотрудника входит в список ключевых позиций: 'Администратор', 'Техник', 'Киберспорт-тренер' (*p.name\_of\_position IN* ('Администратор', 'Техник', 'Киберспорт-тренер')). Данный запрос позволяет руководству клуба оперативно получать данные о квалифицированном персонале, отвечающем за основные направления деятельности: управление клиентами, техническое обслуживание оборудования и тренировку киберспортсменов. Запрос и его результат выполнения представлен на рисунке 19.

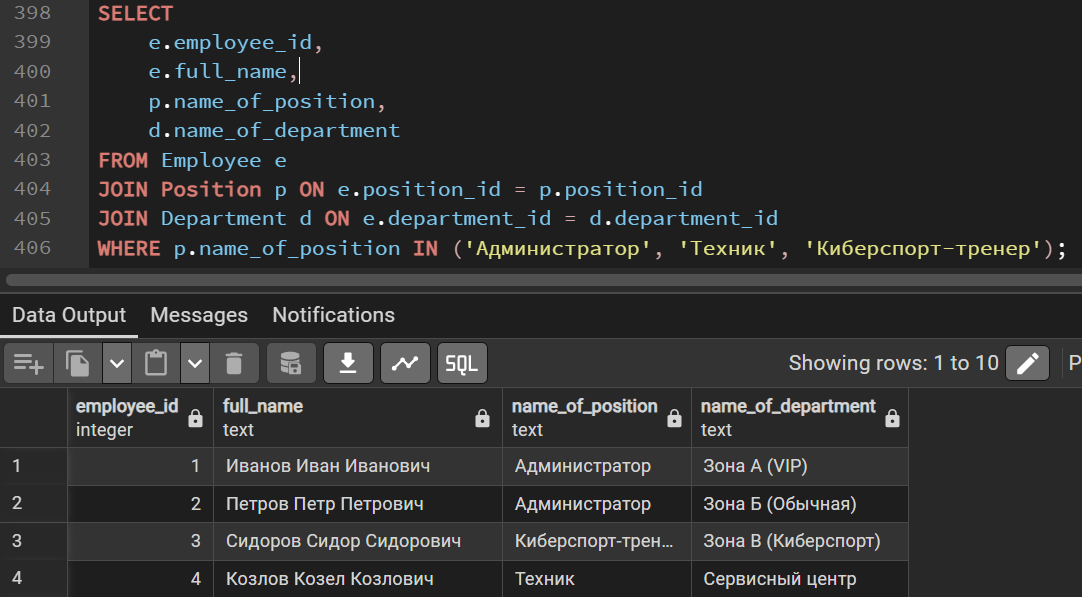


Рисунок 19 – Многотабличный запрос, с оператором in

Многотабличный запрос с оператором *not in*. Создадим запрос для анализа основного игрового оборудования, исключая вспомогательные системы. Запрос объединяет таблицы *ComputingEquipment* и *EquipmentCategory*, выбирая записи, где категория оборудования не входит в список технических категорий: 'Сетевое оборудование', 'Серверное оборудование' (*ec.name\_of\_category NOT IN* ('Сетевое оборудование', 'Серверное оборудование')). Данный запрос позволяет руководству компьютерного клуба сосредоточиться на анализе игровых ПК, консолей и периферии, исключая из отчетов инфраструктурное оборудование, что помогает в оценке основного игрового парка и планировании закупок для клиентских зон. Запрос и его результат выполнения представлен на рисунке 20.

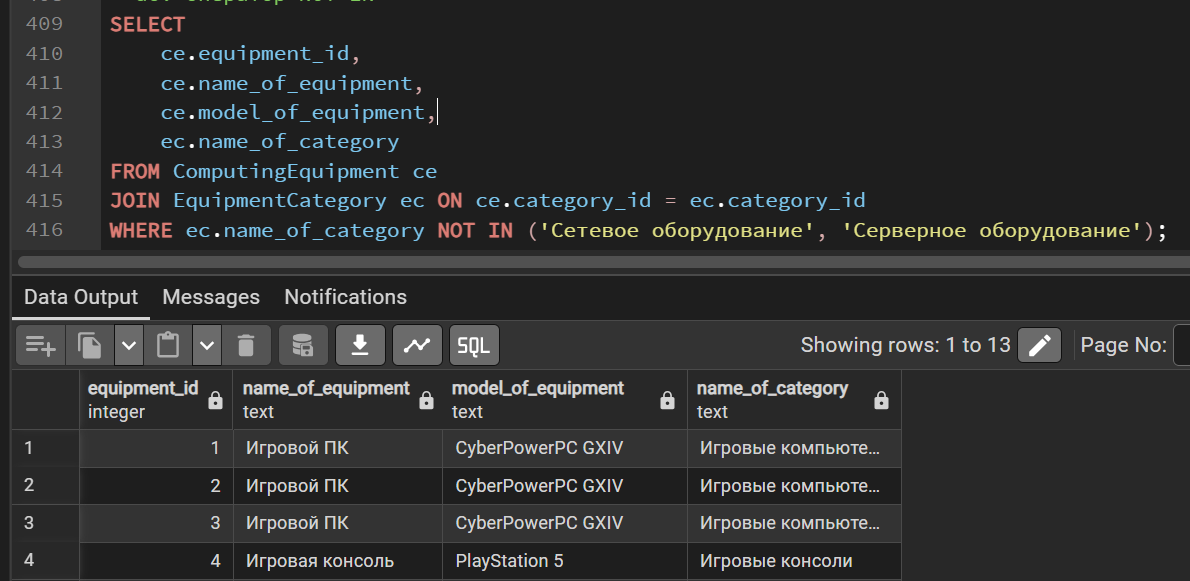


Рисунок 20 – Многотабличный запрос, с оператором not in

# **5.2 Запросы с оконными функциями**

Запрос с оконной функцией *sum() over*. Создадим запрос для анализа загруженности сотрудников перемещениями оборудования в их подразделениях. Запрос объединяет таблицы *Movement, Employee* и *Department,* рассчитывая для каждого сотрудника количество выполненных операций перемещения (*COUNT()),* а также используя оконную функцию *SUM(COUNT()) OVER (PARTITION BY d.department\_id*) для определения общего количества перемещений по всему подразделению. Это позволяет сравнить индивидуальные показатели каждого сотрудника с общими результатами его отдела. Данный запрос помогает руководству компьютерного клуба оценить распределение рабочей нагрузки между сотрудниками в разных зонах и выявить наиболее активных специалистов для поощрения или анализа опыта работы. Запрос и его результат выполнения представлен на рисунке 21.

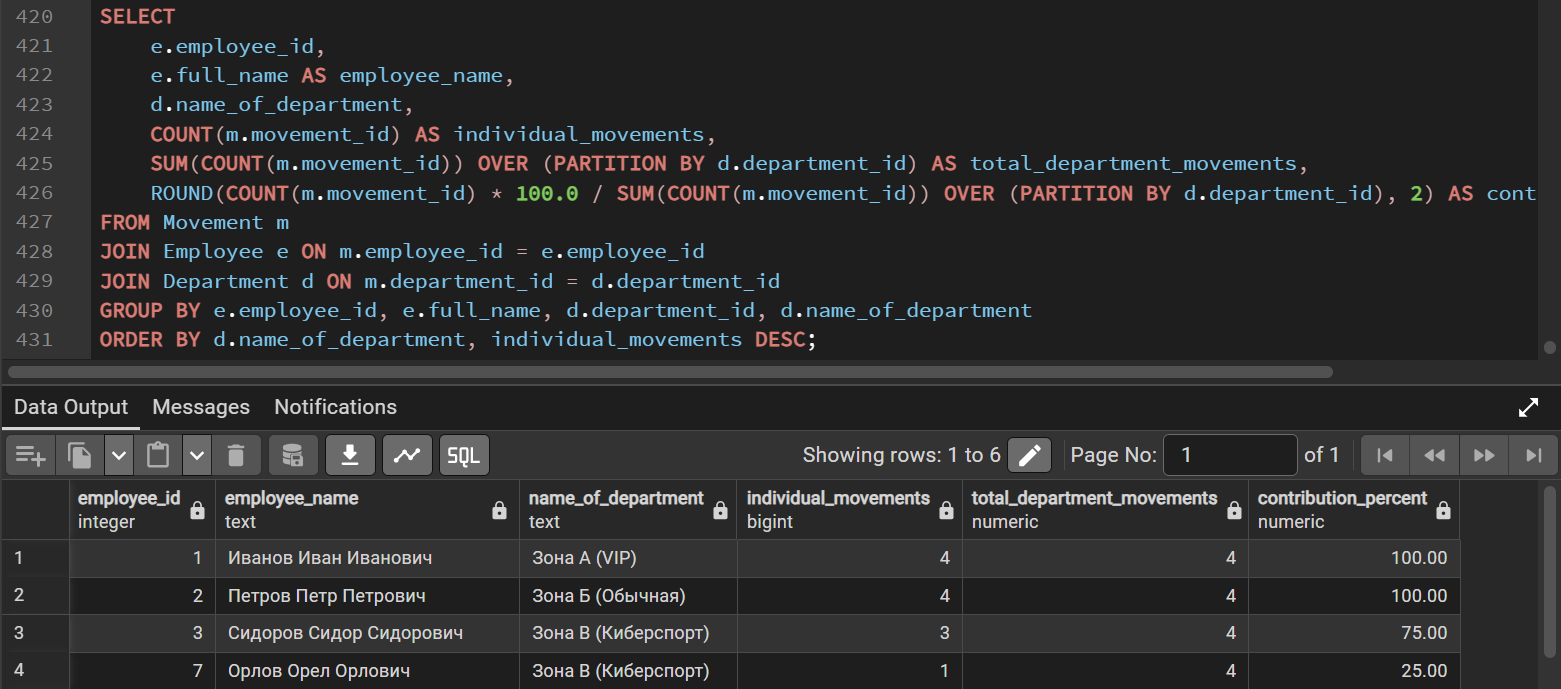


Рисунок 21 – Запрос с оконной функцией sum() over

Запрос с оконной функцией *row\_number() over.* Создадим запрос для отслеживания последних операций с оборудованием в каждом подразделении компьютерного клуба. Запрос объединяет таблицы *Movement, InventoryNumber, ComputingEquipment, Department* и *MovementType,* группируя данные по подразделениям *(department\_id)* и сортируя перемещения внутри каждой группы по дате в порядке убывания (*ORDER BY m.movement\_date DESC*). Оконная функция *ROW\_NUMBER()* присваивает каждой записи порядковый номер в пределах своей группы, где *rn* = 1 соответствует самой последней операции с оборудованием в данном подразделении. Данный запрос позволяет администратору клуба быстро идентифицировать актуальное расположение оборудования и недавние перемещения в каждой игровой зоне, что критически важно для оперативного управления техникой и реагирования на запросы клиентов. Для получения только самых последних перемещений можно обернуть запрос в подзапрос и добавить условие *WHERE rn* = 1. Запрос и его результат выполнения представлен на рисунке 22.

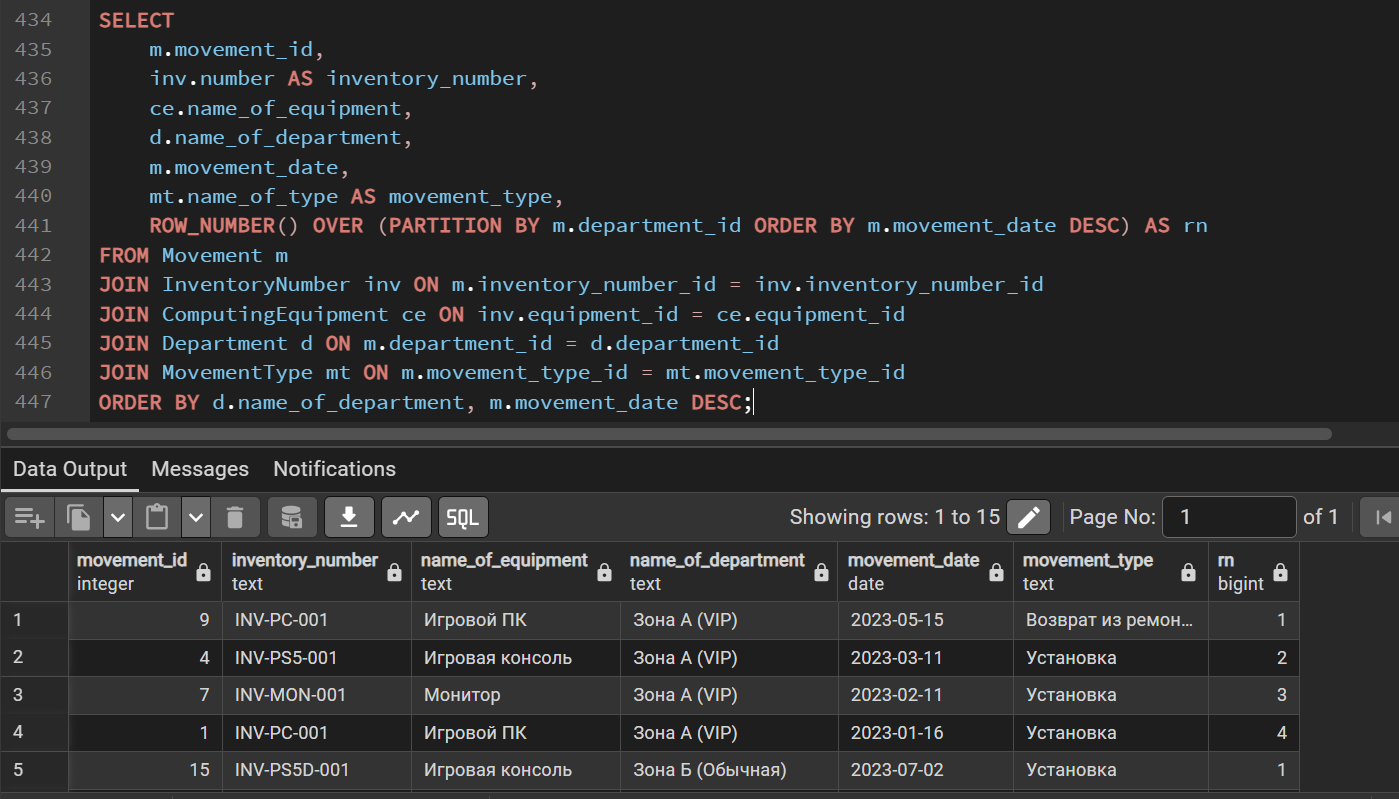


Рисунок 22 – Запрос с оконной функцией row\_number() over

Запрос с оконной функцией *ntile() over.* Создадим запрос для ранжирования игровых зон компьютерного клуба по активности перемещения оборудования. Запрос объединяет таблицы *Movement* и *Department,* группирует данные по подразделениям, подсчитывает общее количество операций перемещения оборудования для каждой зоны и распределяет их на четыре равные группы (квартили) с помощью функции *NTILE(4).* Сортировка выполняется по убыванию количества перемещений (*ORDER BY COUNT(m.movement\_id) DESC*), что позволяет:

* 1-й квартиль – зоны с наибольшей активностью перемещений (*VIP*-зона и киберспортивная арена),
* 2-3-й квартили – зоны со средней активностью,
* 4-й квартиль – зоны с наименьшей активностью (сервисный центр и складские помещения).

Такое распределение помогает руководству клуба быстро выявлять наиболее динамичные игровые зоны, требующие повышенного внимания к обслуживанию оборудования, а также анализировать нагрузку на персонал для оптимального распределения ресурсов. Запрос и его результат выполнения представлен на рисунке 23.

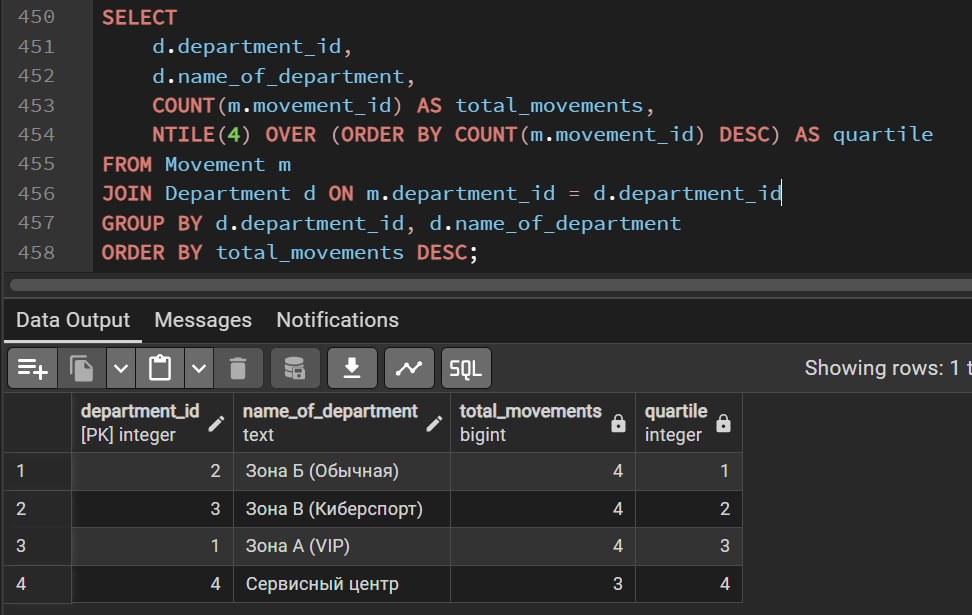


Рисунок 23 – Запрос с оконной функцией ntile() over

# **5.3 Запросы с вычисляемым полем**

Запрос с вычисляемым полем. Создадим запрос, который формирует вычисляемое поле *days\_until\_maintenance,* отражающее количество дней до следующего планового технического обслуживания оборудования. Запрос объединяет таблицы *RegisterOfRepairWorks, InventoryNumber, ComputingEquipment* и *EquipmentCondition,* рассчитывая разницу между текущей датой и датой последнего обслуживания (*CURRENT\_DATE* - *rrw.last\_update AS days\_in\_current\_state*), а также определяя статус внимания с помощью условного выражения *CASE.* Это позволяет сервисной службе компьютерного клуба оперативно выявлять оборудование, требующее немедленного внимания (более 30 дней без обслуживания), и планировать график профилактических работ для предотвращения поломок в пиковые часы работы клуба. Данный запрос является ключевым инструментом в системе предиктивного обслуживания и помогает минимизировать простои игрового оборудования. Запрос и его результат выполнения представлен на рисунке 24.

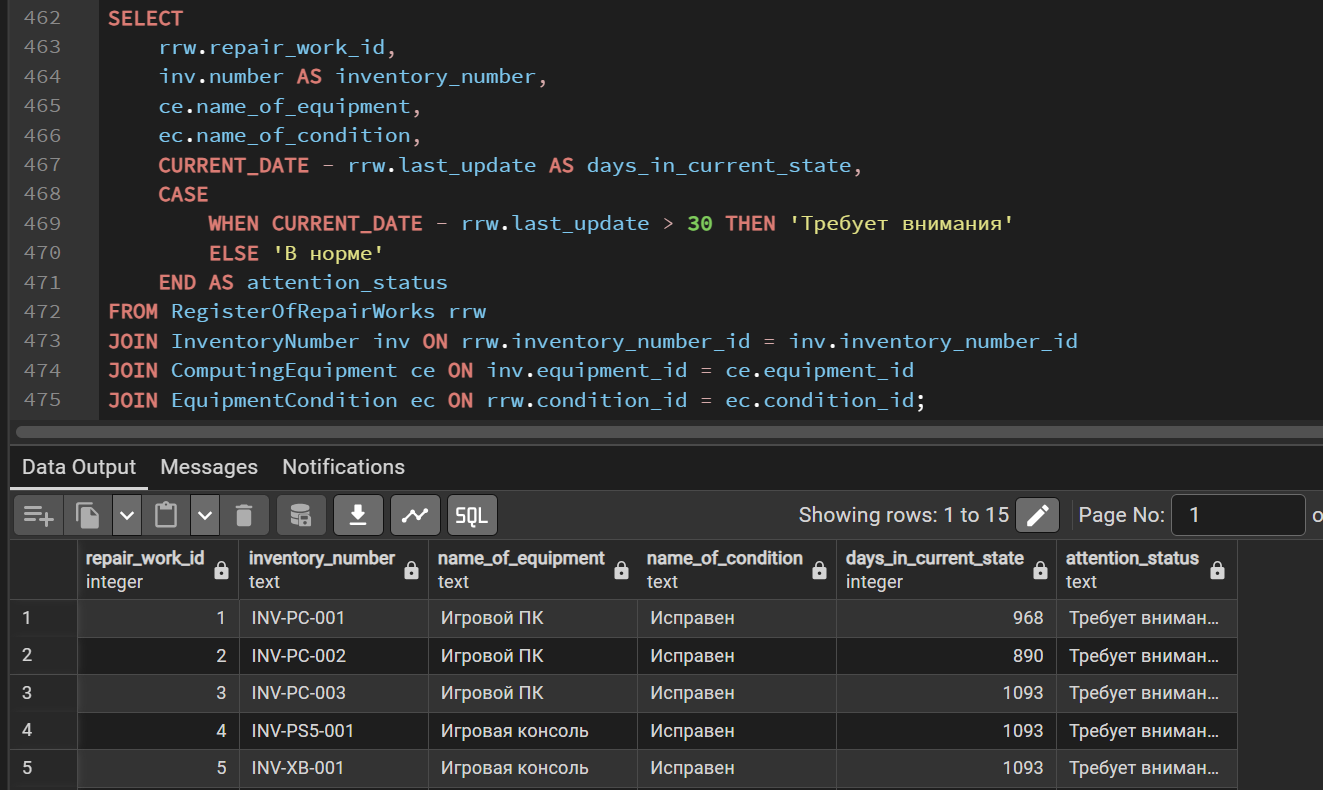


Рисунок 24 – Запрос с вычисляемым полем available\_after\_shipment

Запрос с вычисляемым полем. Создадим запрос, который вычисляет процент компонентов оборудования, требующих замены или ремонта, относительно общего количества компонентов данного типа. Запрос объединяет таблицы *Component, ComputingEquipment, EquipmentCategory* и *InventoryNumber,* группируя данные по типам компонентов и категориям оборудования. Вычисляемое поле *repair\_percentage* показывает долю компонентов со статусом *is\_removable = true* (требующих замены) от общего количества компонентов данного типа. Это позволяет сервисной службе компьютерного клуба оперативно идентифицировать наиболее уязвимые компоненты (например, видеокарты или *SSD*-накопители), которые чаще всего выходят из строя и требуют закупки запасных частей. Данный анализ помогает оптимизировать бюджет на закупку комплектующих и минимизировать время простоя игрового оборудования во время ремонтов. Запрос и его результат выполнения представлен на рисунке 25.

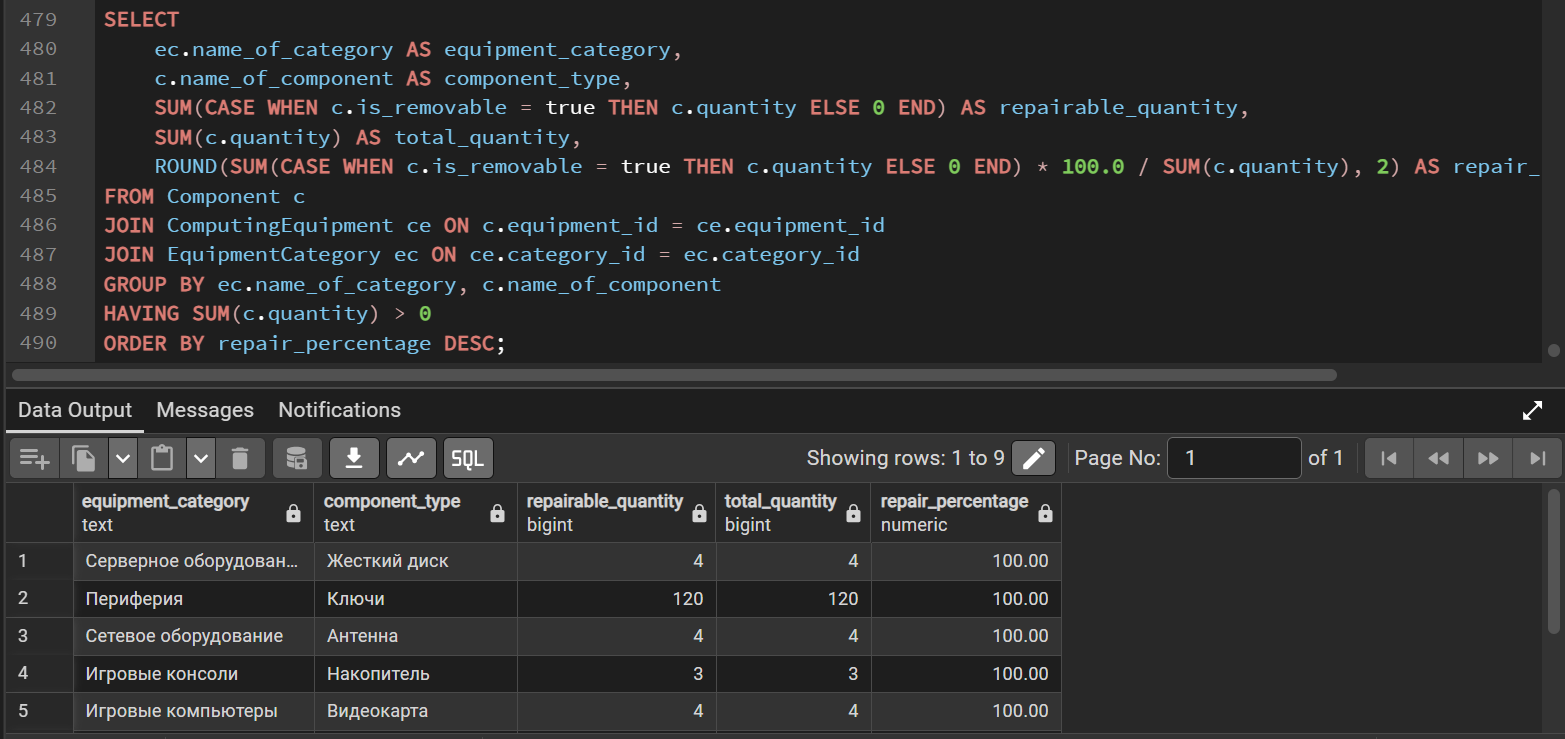


Рисунок 25 – Запрос с вычисляемым полем pct\_of\_stock

# **5.4 Запросы с агрегатными функциями**

Запрос с группировкой данных и простой сортировкой с агрегатной функцией *sum().* Создадим запрос, который группирует данные по категориям оборудования и показывает общее количество компонентов в каждой категории. Запрос объединяет таблицы *Component, ComputingEquipment* и *EquipmentCategory,* используя функцию *SUM(c.quantity)* для подсчета общего числа компонентов в каждой категории. Результаты сортируются по убыванию общего количества компонентов, что позволяет сервисной службе компьютерного клуба быстро идентифицировать категории оборудования, требующие наибольшего внимания при закупке запасных частей и обслуживании. Данный запрос помогает оптимизировать складские запасы комплектующих и планировать бюджет на ремонтные работы. Запрос и его результат выполнения представлен на рисунке 26.

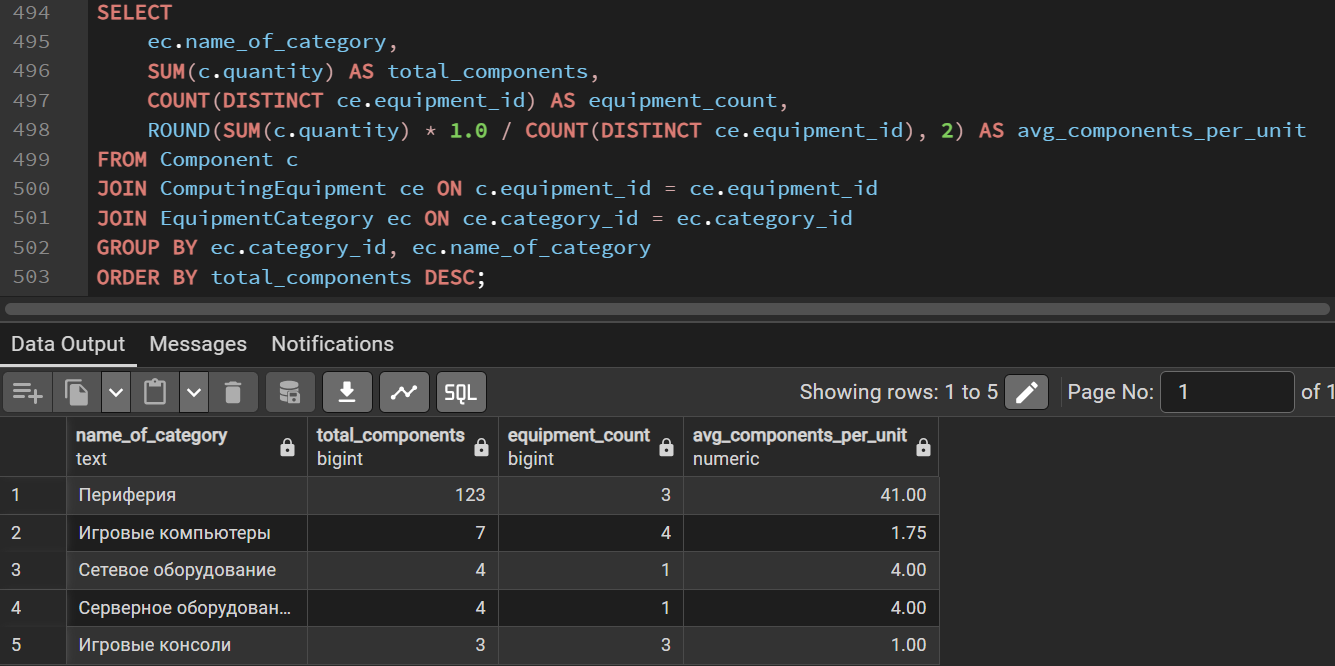


Рисунок 26 – Запрос с группировкой данных и простой сортировкой с агрегатной функцией sum()

Запрос с группировкой данных с условием и многоуровневой сортировкой с агрегатной функцией *sum().* Создадим запрос, который группирует перемещения оборудования по сотрудникам и категориям техники, учитывает только активных специалистов (более 2 перемещений за период) и сортирует данные по нескольким уровням: сначала по общему количеству перемещений, потом по ФИО сотрудника, затем по названию категории оборудования. Запрос объединяет таблицы *Movement, Employee, InventoryNumber, ComputingEquipment* и *EquipmentCategory,* используя *HAVING COUNT(m.movement\_id)* >= 2 для фильтрации на уровне групп. Данный анализ позволяет руководству компьютерного клуба выявить наиболее активных сотрудников в различных категориях оборудования, что помогает в оптимизации распределения обязанностей, планировании обучения персонала и формировании команд для крупных турниров или мероприятий. Запрос и его результат выполнения представлен на рисунке 27.

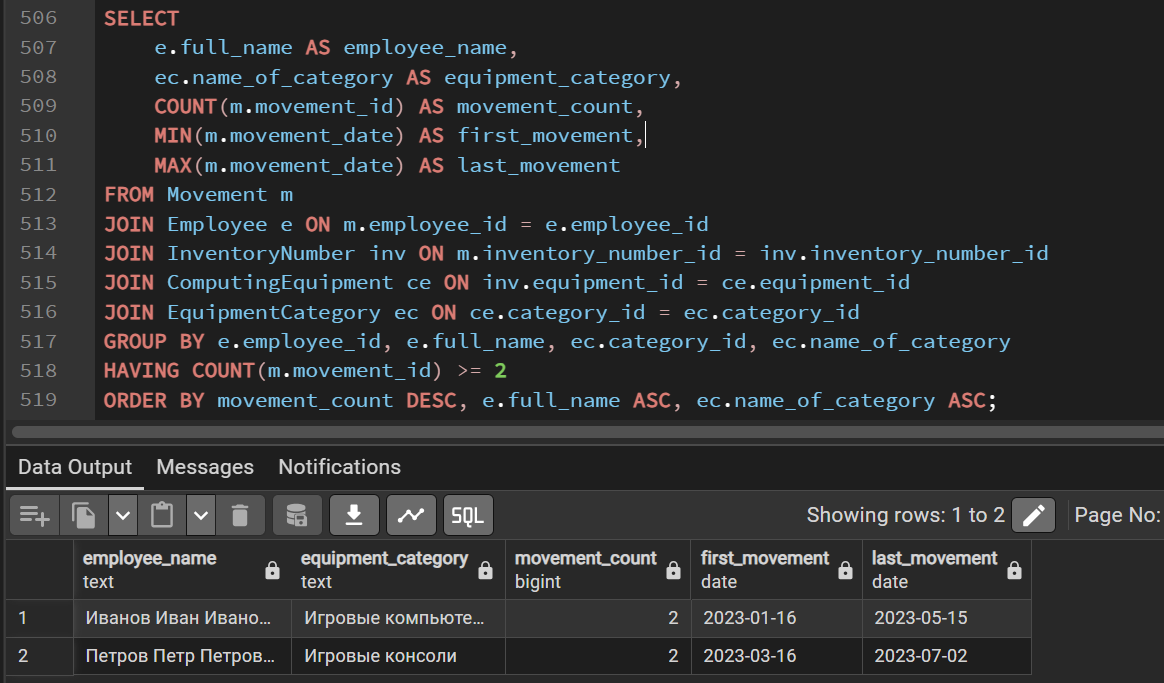


Рисунок 27 – Запрос с группировкой данных и простой сортировкой с агрегатной функцией sum()

# **5.5 Запросы с подзапросами**

Запрос с использованием подзапроса, возвращающего единичное значение. Создадим запрос, который возвращает все компоненты оборудования, количество которых превышает среднее количество компонентов по всей базе данных. Здесь подзапрос вычисляет единственное значение – среднее количество компонентов *(SELECT AVG(quantity) FROM Component),* а основной запрос фильтрует записи по условию *quantity* > среднее\_значение. Данный запрос позволяет сервисной службе компьютерного клуба оперативно выявлять компоненты с аномально высоким количеством (например, механические клавиши в клавиатурах или кабели питания), что помогает в планировании закупок запасных частей, оптимизации складских запасов и прогнозировании износа оборудования. Особое внимание уделяется деталям, требующим частой замены в процессе эксплуатации игровых станций. Запрос и его результат выполнения представлен на рисунке 28.

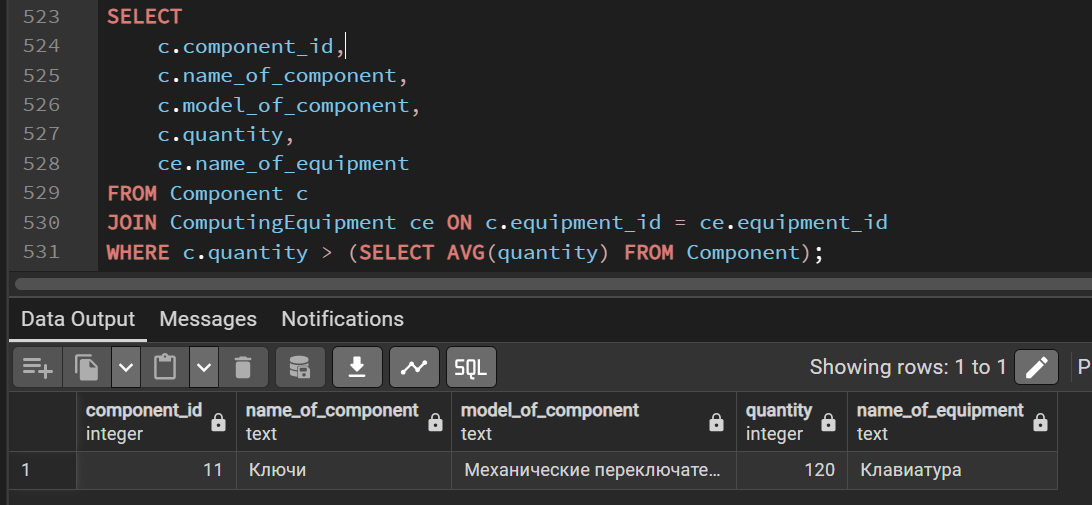


Рисунок 28 – Запрос с использованием подзапроса, возвращающего единичное значение

Запрос с использованием операций *in* и *not in*. Составим запрос, который показывает все подразделения, где работает персонал в ключевых должностях компьютерного клуба, и исключает подразделения, где есть сотрудники на позиции "Уборщик". Запрос объединяет таблицы *Employee,* *Position* и *Department,* используя операцию *IN* для фильтрации подразделений с названиями, содержащими подстроку '%он%' (например, "Зона А (*VIP*)", "Зона В (Киберспорт)", "Сервисный центр"), и операцию NOT IN для исключения записей о сотрудниках, занимающих должность уборщика. Данный запрос позволяет руководству клуба анализировать распределение основного персонала (администраторов, техников, тренеров) по игровым зонам, исключая вспомогательный персонал из отчета для фокусировки на ключевых операционных процессах. Запрос и его результат выполнения представлен на рисунке 29.

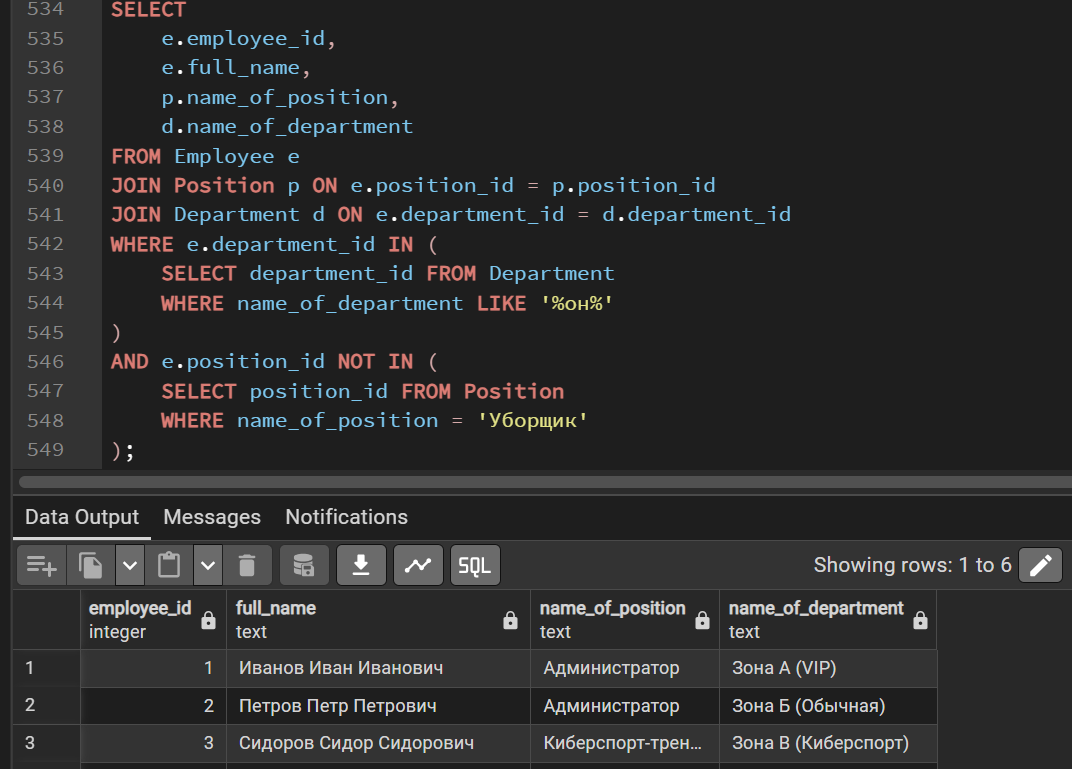


Рисунок 29 – Запрос с использованием операций *in и not*

Запрос с использованием ключевых слов *any* и *all*. Создадим запрос, который показывает все игровые ПК, выпущенные позже всех игровых консолей в базе данных, а также любое оборудование, содержащее видеокарту *NVIDIA GeForce RTX 4090*. Запрос использует оператор *ALL* для сравнения года выпуска оборудования с максимальным годом выпуска среди игровых консолей (*ce.year\_manufactured > ALL (SELECT year\_manufactured FROM ComputingEquipment WHERE category\_id* = ...)), и оператор *ANY* для проверки наличия в оборудовании компонентов с заданной моделью видеокарты (*ce.equipment\_id = ANY (SELECT equipment\_id FROM Component WHERE* ...)). Данный запрос позволяет руководству компьютерного клуба оперативно идентифицировать самое современное оборудование, а также выделять системы с топовыми видеокартами для *VIP*-зон и киберспортивных соревнований. Особую ценность представляет возможность одновременного анализа как временных характеристик оборудования, так и его технических спецификаций в одном запросе. Запрос и его результат выполнения представлен на рисунке 30.

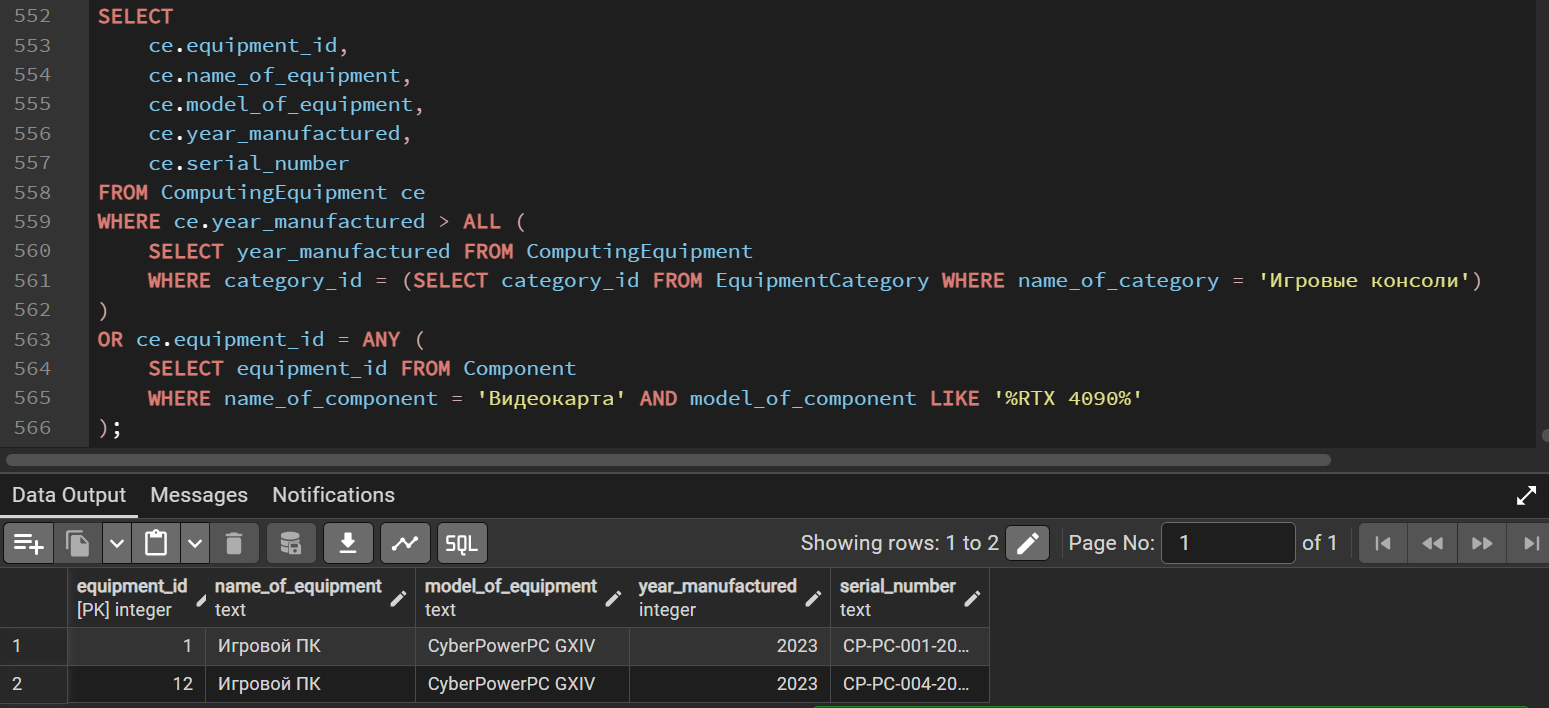


Рисунок 30 – Запрос с использованием ключевых слов *any* и *all*

Запрос с использованием ключевых слов *exists* и *not exists*. Создадим запрос, который возвращает все подразделения компьютерного клуба, в которых были зафиксированы перемещения оборудования за второе полугодие 2023 года, и исключает подразделения, где работает хотя бы один киберспорт-тренер. Запрос объединяет таблицы *Department* и *Organization,* используя подзапрос с *EXISTS* для проверки наличия записей о перемещениях в *Movement* с датой после 1 июня 2023 года (*m.movement\_date* > '2023-06-01'), и подзапрос с *NOT EXISTS* для исключения подразделений, где есть сотрудники с должностью "Киберспорт-тренер". Данный запрос позволяет руководству клуба анализировать активность в "обычных" игровых зонах, отделенных от киберспортивных направлений, что помогает в раздельной оценке нагрузки на оборудование и планировании обновлений для разных типов клиентов. Запрос и его результат выполнения представлен на рисунке 31.

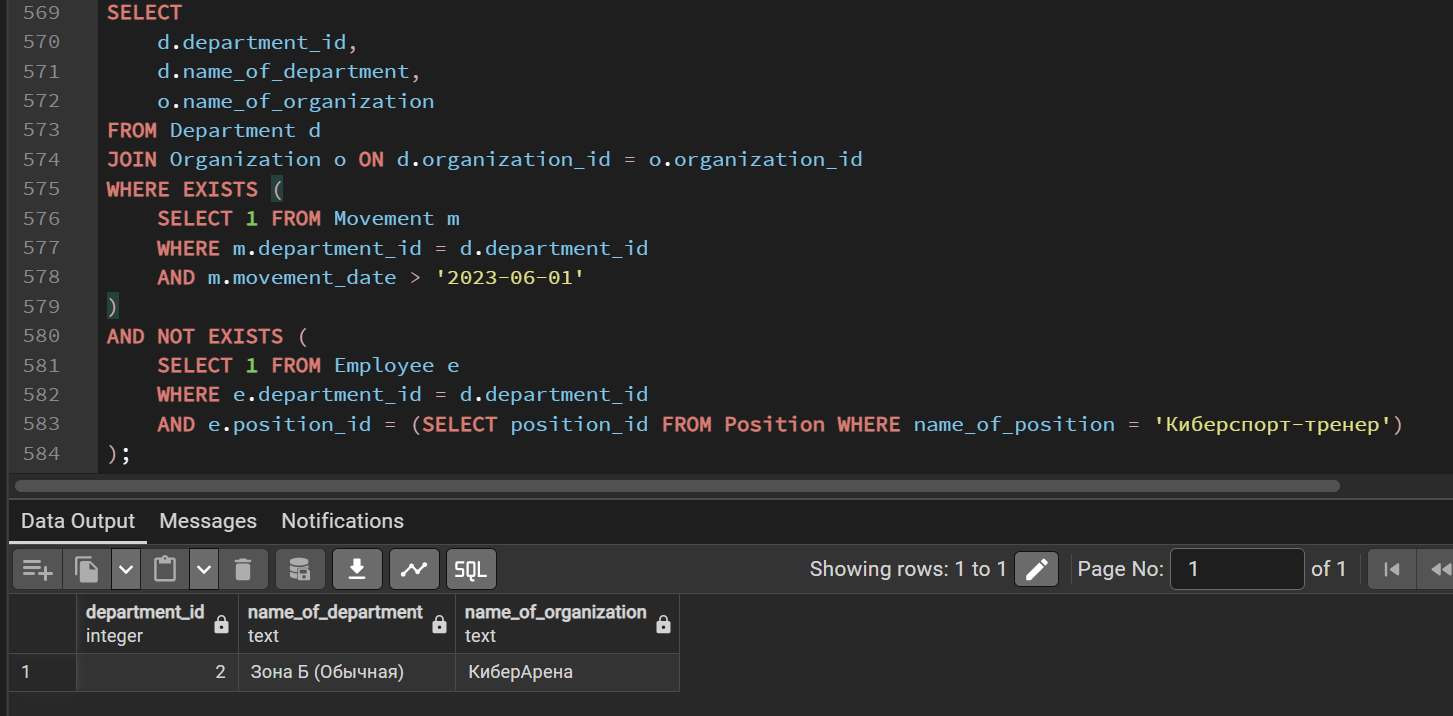


Рисунок 31 – Запрос с использованием ключевых слов *exists* и *not* *exists*

# **5.6 Запрос на основе материализованного представления**

Создадим материализованное представление *mv\_equipment\_summary,* которое суммирует текущее состояние оборудования и статистику перемещений по каждому подразделению и категории техники. Это позволит быстро получать агрегированную информацию о распределении игровых станций, консолей и периферии по зонам компьютерного клуба без необходимости выполнять сложные запросы в реальном времени. Представление объединяет данные из таблиц *RegisterOfRepairWorks, Department, EquipmentCategory, EquipmentCondition* и *ComputingEquipment*, группируя их по подразделениям и категориям оборудования, и вычисляя ключевые метрики: общее количество единиц, количество исправного оборудования, средний возраст техники в годах и дату последнего обслуживания. Данное представление особенно полезно для ежедневного мониторинга состояния парка оборудования и оперативного формирования отчетов для руководства клуба. Запрос для создания материализованного представления отображен на рисунке 32.

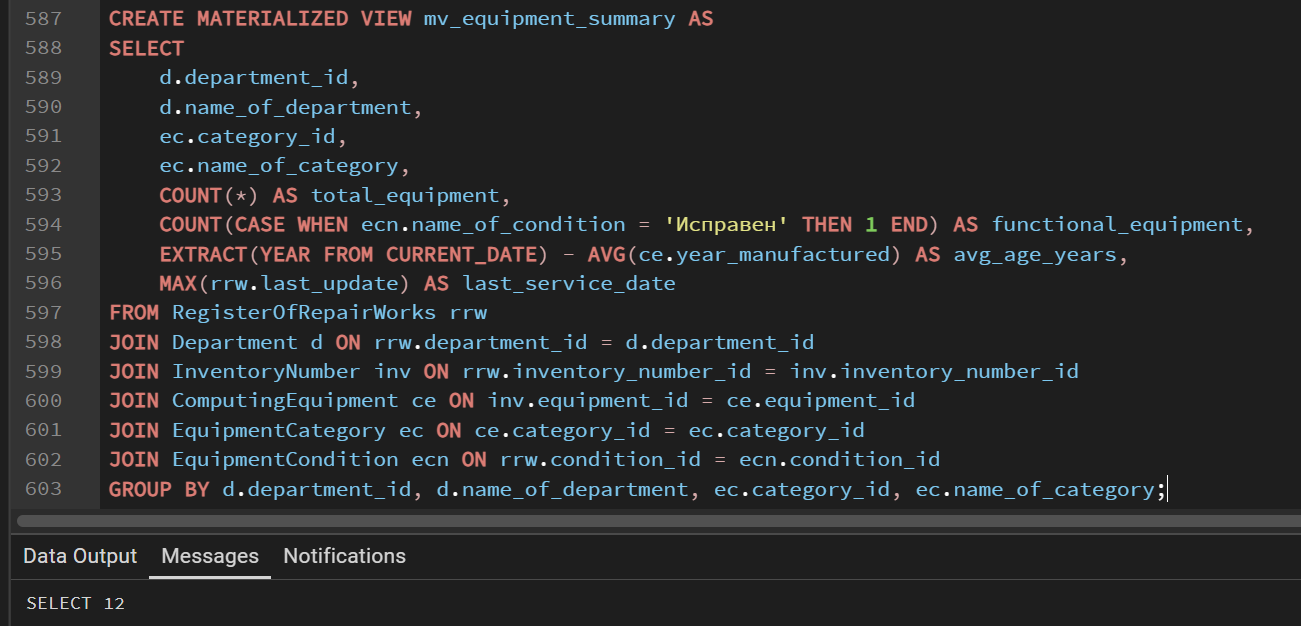


Рисунок 32 – Запрос на создание материализованного представления

Создадим запрос к материализованному представлению *mv\_equipment\_summary,* который покажет категории оборудования в *VIP*-зоне с процентным отношением исправного оборудования к общему количеству единиц. Запрос фильтрует данные по *department\_id* = 1 (Зона А - *VIP*), вычисляет процент работоспособной техники как отношение *functional\_equipment* к *total\_equipment*, и сортирует результаты по убыванию этого показателя. Данный запрос позволяет руководству компьютерного клуба быстро оценить техническое состояние оборудования в премиальной зоне и определить, какие категории техники требуют первоочередного внимания сервисной службы. Особую ценность представляет возможность мгновенной оценки качества обслуживания *VIP*-клиентов в процентах, что удобно для представления отчетов на совещаниях. Запрос и его результат выполнения представлен на рисунке 33.



Рисунок 33 – Запрос на основе материализованного представления

# **6 Разработка объектов БД**

Первое материализованное представление мы уже создали в пункте 4.9. Теперь создадим второе, которое хранит актуальное состояние оборудования с расшифровкой названий подразделений, категорий техники и текущих состояний. Это позволяет получать полностью готовую сводку по оборудованию одним простым запросом без многократных соединений таблиц, значительно ускоряя формирование отчетов для руководства компьютерного клуба. Для актуализации данных после перемещений оборудования или завершения ремонтных работ используется команда *REFRESH MATERIALIZED VIEW mv\_equipment\_status.* Создание материализованного представления представлен на рисунке 34.

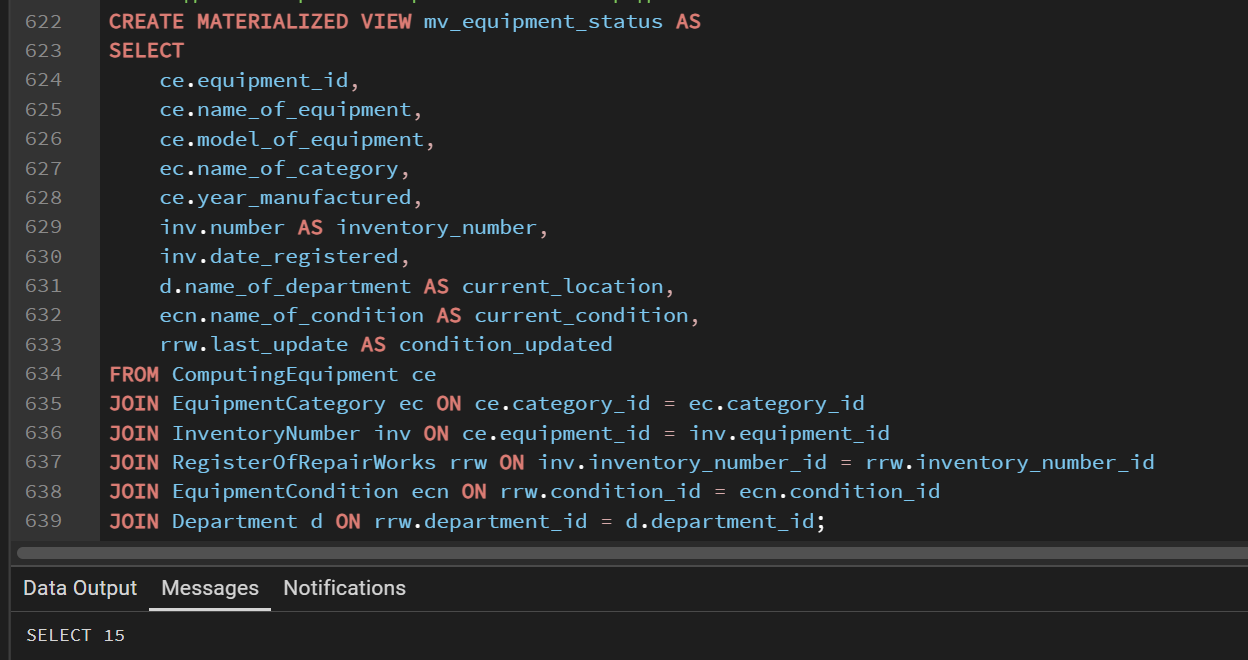


Рисунок 34 – Создание материализованного представления *mv\_equipment\_status*

Создадим запрос к материализованному представлению *mv\_equipment\_status,* который покажет все исправное оборудование в *VIP*-зоне, отсортированное по дате последнего обслуживания. Запрос фильтрует записи по условию *current\_condition* = 'Исправен' и *current\_location* = 'Зона А (*VIP*)', вычисляет возраст оборудования в годах и добавляет текстовый индикатор срочности обслуживания на основе количества дней с последнего обновления состояния. Данный запрос позволяет администратору компьютерного клуба оперативно оценить состояние премиального оборудования и спланировать профилактическое обслуживание для *VIP*-клиентов. Для получения актуальных данных перед выполнением запроса необходимо обновить представление с помощью команды *REFRESH MATERIALIZED VIEW mv\_equipment\_status*. Запрос и его результат выполнения представлен на рисунке 35.

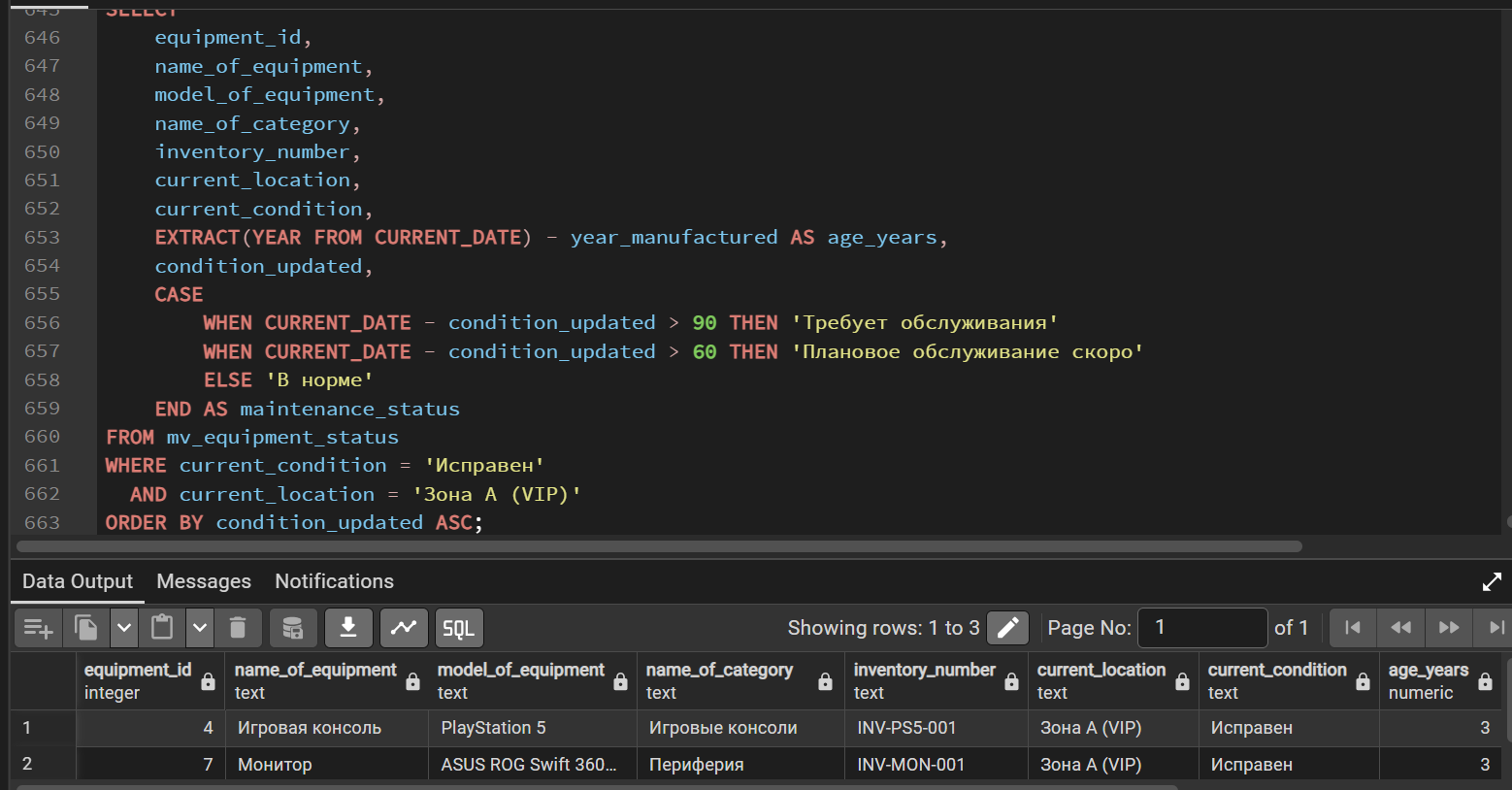


Рисунок 35 – Запрос к *mv\_equipment\_status*

Создадим первый курсор для мониторинга состояния оборудования. Он используется для построчного перебора таблицы *RegisterOfRepairWorks.* Курсор возвращает инвентарный номер оборудования, название подразделения, текущее состояние оборудования и дату последнего обновления состояния. Такой подход позволяет обрабатывать и отображать информацию о состоянии каждой единицы техники в компьютерном клубе, что удобно для анализа текущего состояния парка оборудования и планирования ремонтных работ. Запрос с объявлением и использованием курсора и его результат выполнения представлен на рисунке 36.

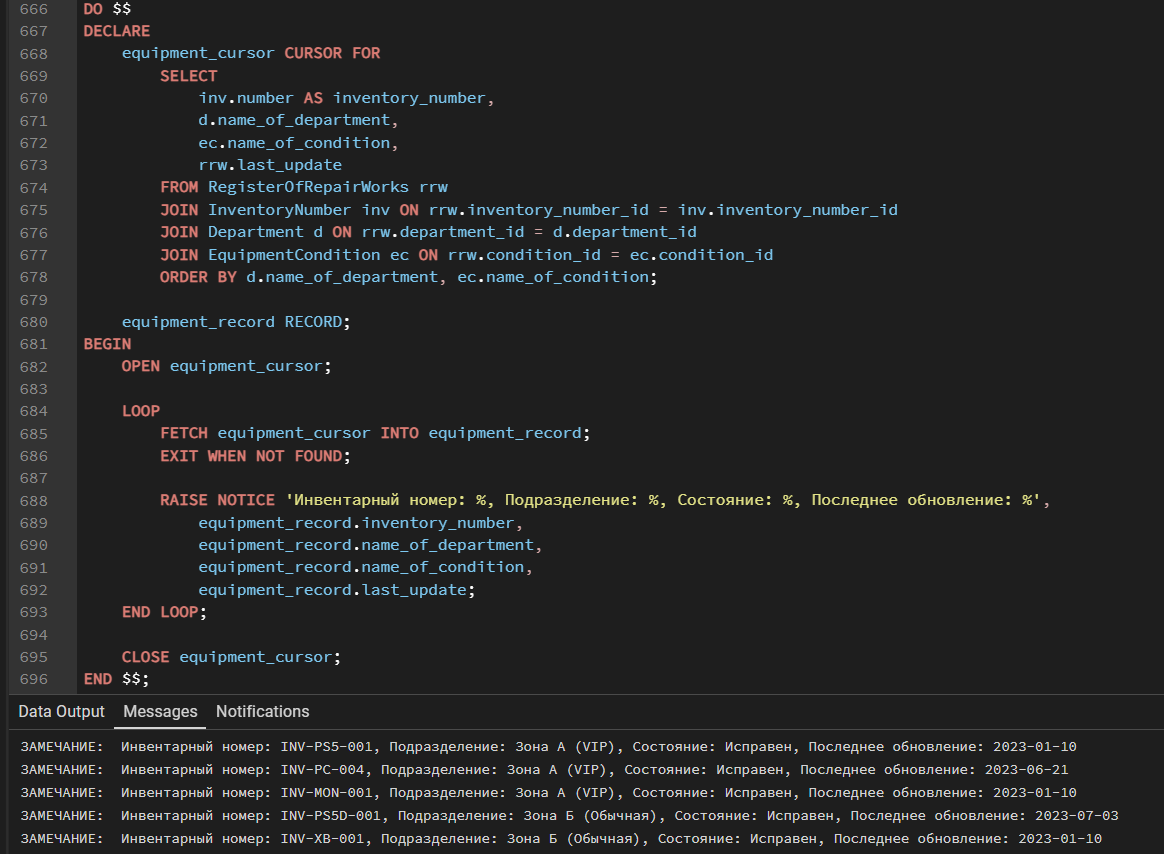


Рисунок 36 – Курсор для мониторинга оборудования

Создадим второй курсор для отслеживания перемещений оборудования. Он используется для построчного перебора таблицы *Movement*. Курсор возвращает идентификатор операции, инвентарный номер оборудования, название подразделения, тип операции и дату перемещения. Это позволяет пошагово анализировать все операции с оборудованием в компьютерном клубе, что полезно для контроля перемещений техники между игровыми зонами, отслеживания ремонтов и апгрейдов, а также для формирования детальных отчетов о работе персонала. Запрос с объявлением и использованием курсора и его результат выполнения представлен на рисунке 37, 38.

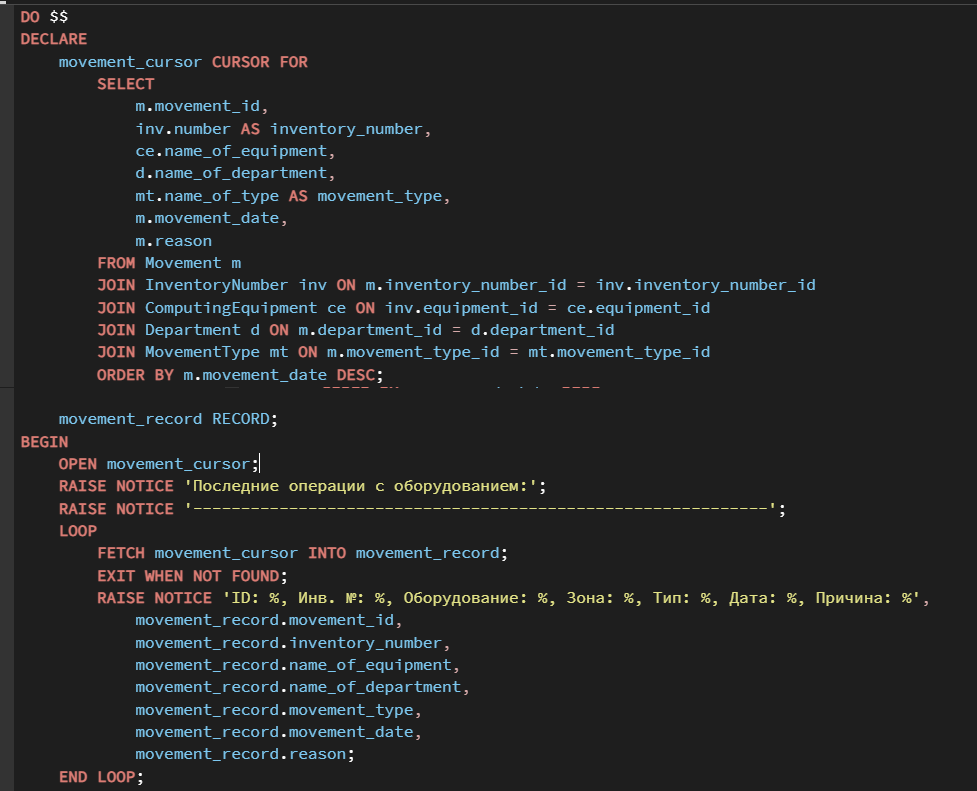


Рисунок 37 – Курсор для отслеживания перемещения

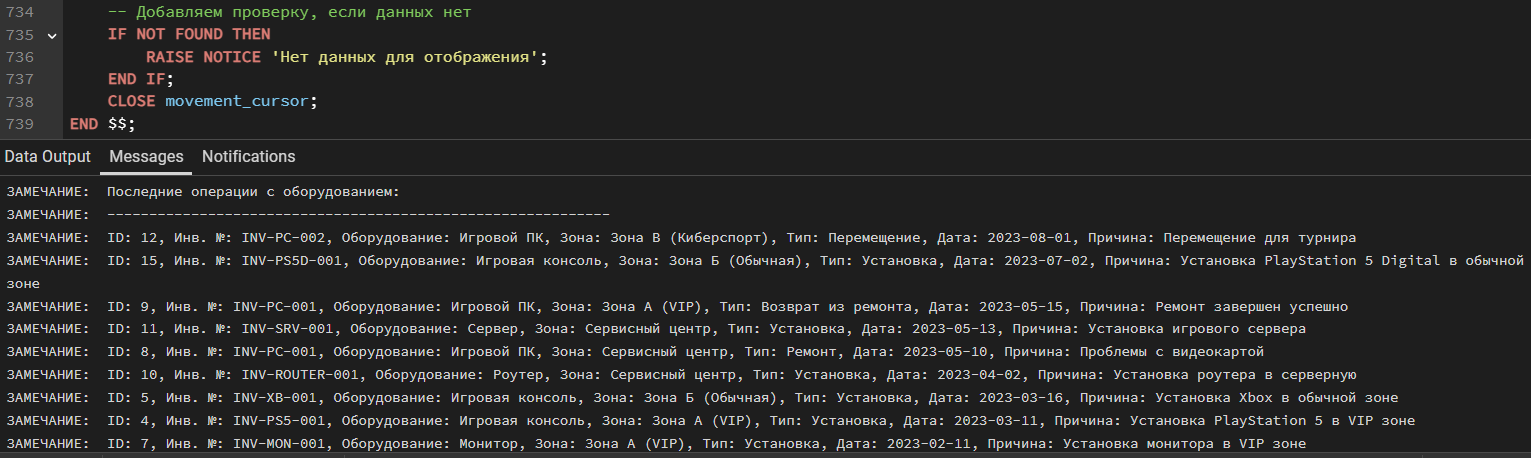


Рисунок 38 – Результат выполнения

Создадим функцию, которая предназначена для получения текущего количества исправного оборудования определенной категории в указанном подразделении компьютерного клуба. Она принимает два параметра: *dept\_id* — идентификатор подразделения, и *cat\_id* — идентификатор категории оборудования, и возвращает значение типа *INTEGER*, соответствующее количеству исправных единиц оборудования в указанной зоне. Данная функция позволяет администратору клуба оперативно получить информацию о доступном для использования оборудовании без выполнения сложных запросов, что особенно полезно при планировании турниров или распределении клиентов по игровым зонам. Запрос на создание функции представлен на рисунке 39.

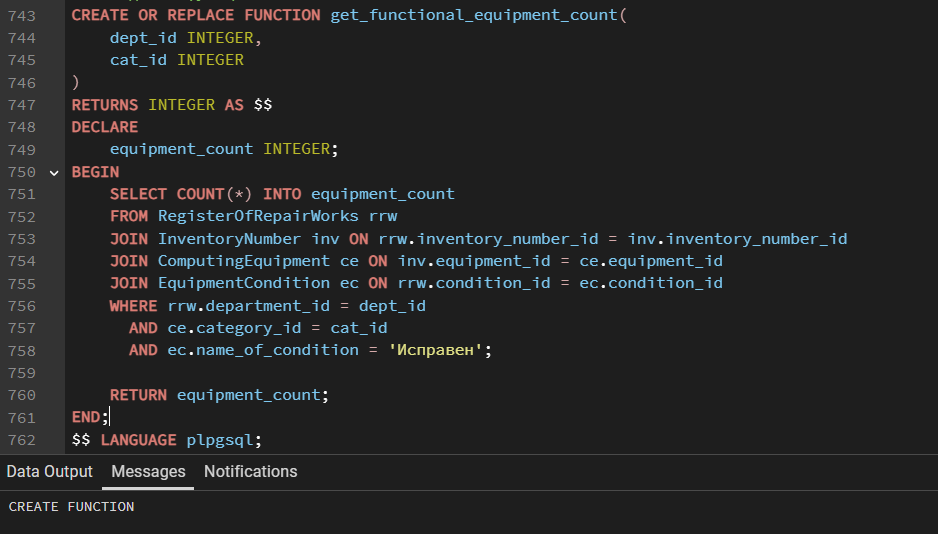


Рисунок 39 – Создание функции *get\_functional\_equipment\_count*

Для проверки работы функции выполняется простой запрос, который возвращает количество исправных игровых ПК (*category\_id* = 1) в *VIP*-зоне (*department\_id* = 1). Функция обеспечивает быстрый доступ к данным о доступном оборудовании и может использоваться в последующих аналитических или отчетных запросах для оперативного планирования загрузки игровых зон и распределения клиентов. Данный подход значительно упрощает процесс принятия решений администратором клуба, позволяя мгновенно получать актуальную информацию о состоянии оборудования без необходимости выполнения сложных многотабличных запросов в реальном времени. Запрос представлен на рисунке 40.

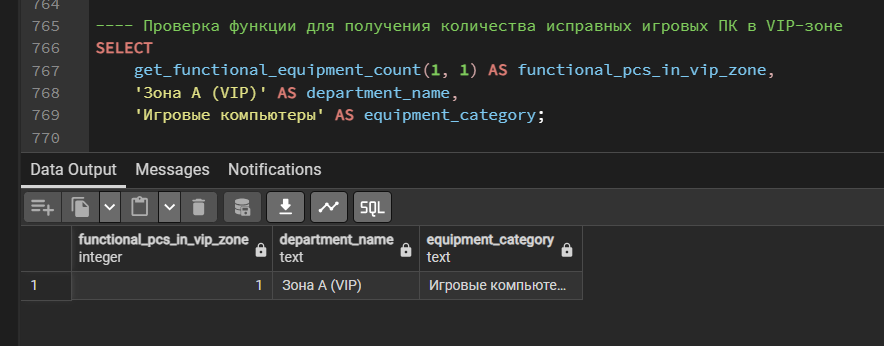


Рисунок 40 – Запрос на использование функции *get\_functional\_equipment\_count*

Создадим процедуру, которая предназначена для регистрации перемещения оборудования между подразделениями компьютерного клуба. Она принимает пять параметров: *p\_inventory\_number\_id* – идентификатор инвентарного номера оборудования, *p\_department\_id* – идентификатор подразделения назначения, *p\_employee\_id* – идентификатор сотрудника, выполняющего перемещение, *p\_movement\_type\_id* – идентификатор типа операции, и *p\_reason* – текстовое описание причины перемещения. При вызове процедуры в таблицу *Movement* добавляется новая запись с текущей датой, а также автоматически обновляется состояние оборудования в реестре ремонтных работ (*RegisterOfRepairWorks*), если перемещение связано с ремонтом или возвратом из ремонта. Данная процедура обеспечивает целостность данных и автоматизирует процесс учёта перемещений техники. Запрос на создание процедуры представлен на рисунке 41.



Рисунок 41 – Запрос на создание процедуры

Для проверки работы процедуры выполняется простой вызов с конкретными значениями параметров. Процедура *register\_equipment\_movement* обеспечивает безопасное и корректное обновление данных о перемещении оборудования между подразделениями компьютерного клуба. В примере происходит перемещение наушников *SteelSeries Arctis Pro* (инвентарный номер *INV-HEAD-001*) из обычной зоны в сервисный центр для ремонта микрофона. Сотрудником, выполняющим операцию, назначен техник Козлов К.К. После вызова процедуры автоматически создаётся запись в журнале перемещений и обновляется состояние оборудования в реестре ремонтных работ. Для визуальной проверки корректности работы процедуры выполняется запрос к таблице *Movement* для просмотра последних операций и к таблице *RegisterOfRepairWorks* для подтверждения изменения состояния оборудования. Запрос представлен на рисунке 42

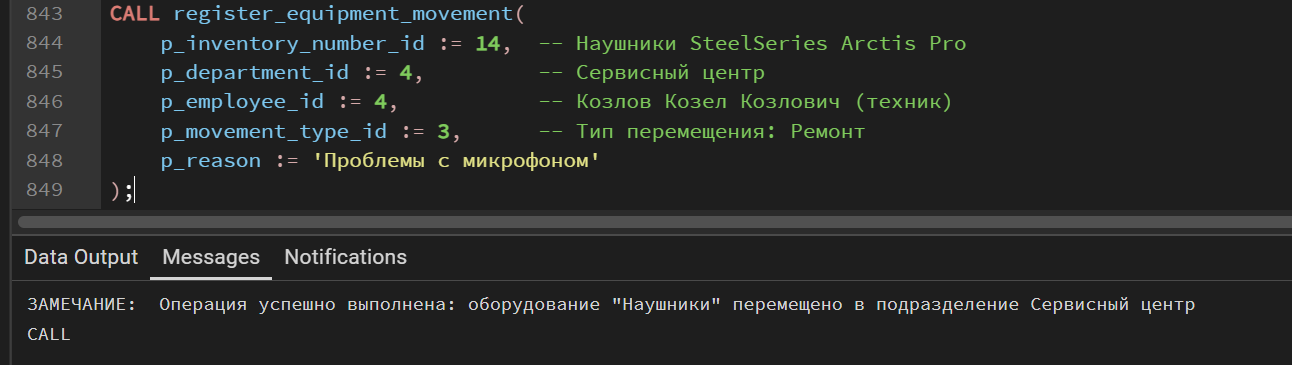


Рисунок 42 – Запрос на использование процедуры

Создадим триггер *trg\_update\_repair\_date,* который автоматически обновляет поле *last\_update* в таблице *RegisterOfRepairWorks* при изменении состояния оборудования *(condition\_id)* или его местоположения *(department\_id).* Для этого используется функция *update\_repair\_date(),* устанавливающая текущую дату в поле *last\_update* при каждом обновлении записи. Этот триггер обеспечивает актуальность информации о состоянии оборудования и позволяет точно отслеживать дату последнего обслуживания или перемещения без необходимости ручного обновления. Это особенно важно для сервисной службы компьютерного клуба, так как позволяет оперативно выявлять оборудование, которое давно не проходило техническое обслуживание, и планировать профилактические работы. Запрос представлен на рисунке 43.

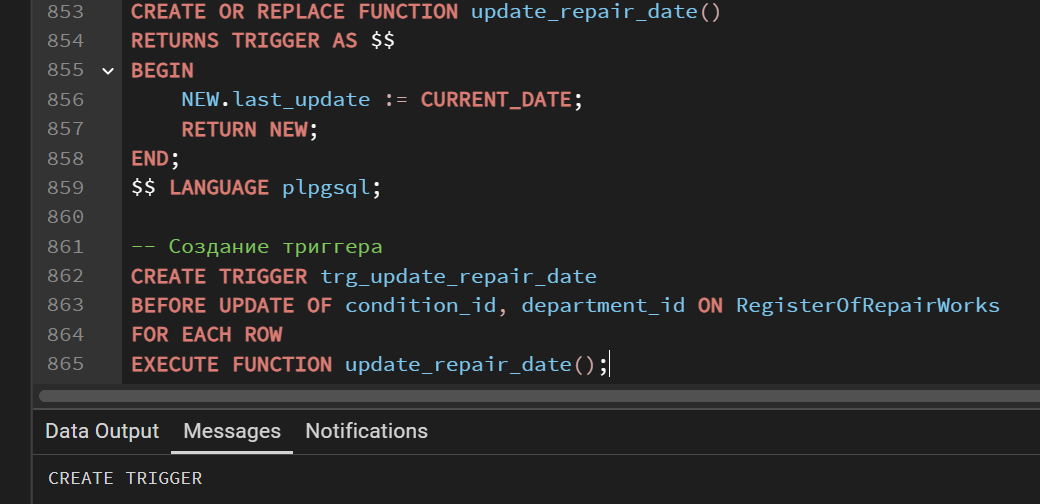


Рисунок 43 – Запрос на создание триггера *trg\_update\_repair\_date*

Создадим триггер *trg\_check\_component\_quantity,* который предотвращает добавление или изменение записи в таблице *Component*, если значение *quantity* меньше или равно нулю. Для этого используется функция *check\_component\_quantity(),* которая выдаёт исключение с подробным сообщением об ошибке при попытке вставить некорректное количество комплектующих. Этот триггер обеспечивает семантическую целостность данных на уровне бизнес-логики, дополняя существующее *CHECK-*ограничение и предотвращая ситуации, когда в системе могут появиться записи с отрицательным или нулевым количеством компонентов, что нарушит расчёт стоимости ремонта, апгрейдов и отчётность по складским запасам. Особую важность это имеет при массовом импорте данных или интеграции с внешними системами, где стандартные ограничения могут быть обойдены. Запрос представлен на рисунке 44.

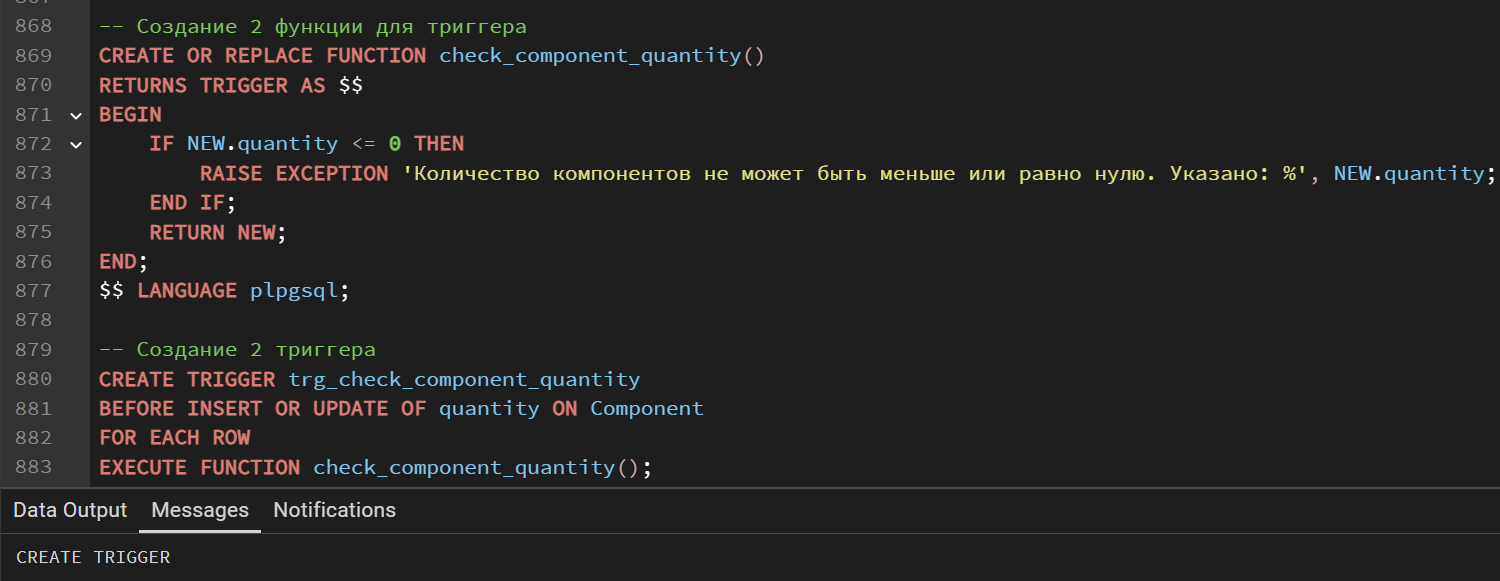


Рисунок 44 – Запрос на создание триггера *trg\_check\_component\_quantity*

# ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была разработана база данных автоматизированного учёта средств вычислительной техники *ComputerInventory*, реализованная на СУБД *PostgreSQL.* База данных обеспечивает хранение и обработку информации об оборудовании компьютерного клуба "КиберАрена", включая игровые ПК, консоли, периферию, комплектующие, инвентарные номера, подразделения, сотрудников, операции перемещения и ремонтные работы, что позволяет эффективно управлять техническим парком, отслеживать состояние оборудования и получать аналитическую информацию для принятия управленческих решений.

В рамках работы были решены следующие задачи:

1. Проведен анализ предметной области компьютерного клуба и определены основные сущности базы данных (оборудование, подразделения, сотрудники, перемещения, ремонты), их атрибуты и взаимосвязи;
2. Спроектированы логическая и физическая модели данных в нотации *IDEF1X* с учетом особенностей учета игрового оборудования, включая связи между инвентарными номерами, комплектующими и историей перемещений;
3. Реализована база данных в *PostgreSQL* с соблюдением ссылочной и семантической целостности: определены первичные и внешние ключи с правилами каскадного обновления и удаления, созданы ограничения *CHECK* (год выпуска > 1980, количество комплектующих ≥ 0), *NOT* *NULL* и *UNIQUE* для критически важных полей;
4. Созданы индексы для повышения производительности запросов к часто используемым полям (даты перемещений, категории оборудования), реализованы материализованные представления для быстрого получения сводной информации, функции для расчета стоимости ремонта, процедуры для регистрации перемещений оборудования и триггеры для автоматического контроля корректности данных;
5. Разработаны аналитические *SQL*-запросы для мониторинга состояния оборудования, отслеживания ремонтных работ, анализа эффективности сотрудников и распределения техники по игровым зонам.

В целом, работа выполнена успешно. База данных функционирует корректно, обеспечивает целостность данных при всех операциях и полностью соответствует требованиям автоматизации учёта средств вычислительной техники в компьютерном клубе "КиберАрена". Реализованные программные объекты значительно упрощают повседневную эксплуатацию системы: от оперативного поиска свободного оборудования для клиентов до формирования отчетов по износу комплектующих и планирования бюджета на ремонтные работы. Предложенное решение может быть легко масштабировано для сети клубов и интегрировано с другими информационными системами учёта.