Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Направление подготовки 11.03.02

Практическая работа №2

Выполнил:

Дощенников Никита Андреевич

Группа: К3221

Проверила:

Татьяна Евгеньевна Войтюк

Санкт-Петербург
2025

Цель работы

Изучение и практическое освоение основных конструкций языка SQL для работы с реляционными базами данных, включая:

- Формирование запросов на выборку данных с использованием оператора `SELECT`
- Применение предложений `WHERE` и `ORDER BY` для фильтрации и сортировки данных
- Использование встроенных функций SQL для обработки числовых, символьных данных и дат
- Работу с условными выражениями для формирования вычисляемых полей
- Получение навыков создания сложных запросов с применением различных операторов и функций SQL

Задачи, решаемые при выполнении работы

- Освоить базовые операции с таблицами:
  - Изучить структуру таблиц базы данных
  - Выполнить выборку данных из таблиц с использованием оператора SELECT
  - Научиться задавать псевдонимы для столбцов
  - Освоить сортировку данных с помощью предложения ORDER BY
- Научиться фильтровать и ограничивать выборку данных:
  - Применять предложение WHERE для фильтрации строк по различным условиям
  - Использовать операторы сравнения и логические операторы
  - Работать с диапазонами значений и списками
  - Выполнять выборку уникальных значений с помощью DISTINCT
- Освоить работу с числовыми функциями:
  - Применять функции округления
  - Выполнять арифметические операции над числовыми данными
  - Создавать вычисляемые поля в результатах запросов
- Изучить символьные функции:
  - Использовать функции конкатенации строк
  - Применять функции изменения регистра
  - Работать с функциями определения длины строк
  - Использовать функции поиска позиции символов
- Освоить функции работы с датами:
  - Выполнять операции с датами и временем
  - Вычислять разницу между датами
  - Извлекать компоненты дат
  - Применять функции преобразования типов данных
- Научиться использовать условные выражения:
  - Применять функцию COALESCE для обработки NULL-значений
  - Использовать конструкцию CASE для создания условной логики в запросах
  - Формировать сложные вычисляемые поля с использованием условий
- Развить навыки анализа и отладки SQL-запросов:
  - Находить и исправлять синтаксические ошибки в запросах
  - Оптимизировать структуру запросов для получения требуемых результатов

Исходные данные

Для выполнения практической работы используется учебная база данных `EmployeesDepartments`, содержащая информацию о сотрудниках компании и структуре отделов.

1. Таблица `EMPLOYEES`

Содержит информацию о сотрудниках компании со следующими полями:

- `EMPLOYEE_ID` – идентификационный номер сотрудника (числовой)
- `FIRST_NAME` – имя сотрудника (строковый)
- `LAST_NAME` – фамилия сотрудника (строковый)
- `EMAIL` – электронная почта (строковый)
- `PHONE_NUMBER` – номер телефона (строковый)
- `HIRE_DATE` – дата найма (дата)
- `JOB_ID` – идентификатор должности (строковый)
- `SALARY` – оклад сотрудника (числовой)
- `COMMISSION_PCT` – процент комиссионных (числовой)
- `MANAGER_ID` – идентификатор менеджера (числовой)
- `DEPARTMENT_ID` – идентификатор отдела (числовой)

2. Таблица DEPARTMENTS

Содержит информацию об отделах компании со следующими полями:

- `DEPARTMENT_ID` – идентификационный номер отдела (числовой)
- `DEPARTMENT_NAME` – название отдела (строковый)
- `MANAGER_ID` – идентификатор руководителя отдела (числовой)
- `LOCATION_ID` – идентификатор местоположения (числовой)

3. Таблица JOB_GRADES

Содержит информацию о разрядах различных должностей.

Выполнение работы

Задание 1. Описание структуры таблицы, выборка данных из таблицы, задание имен столбцов, сортировка строк с помощью предложения ORDER BY

1.1 Будет ли успешна эта команда `SELECT`?

```
SELECT *
FROM "EmployeesDepartments".JOB_GRADES;
```

Нет, не будет, так как для корректного выполнения данной команды, нужно добавить кавычки у `JOB_GRADES` (Рис. 1).

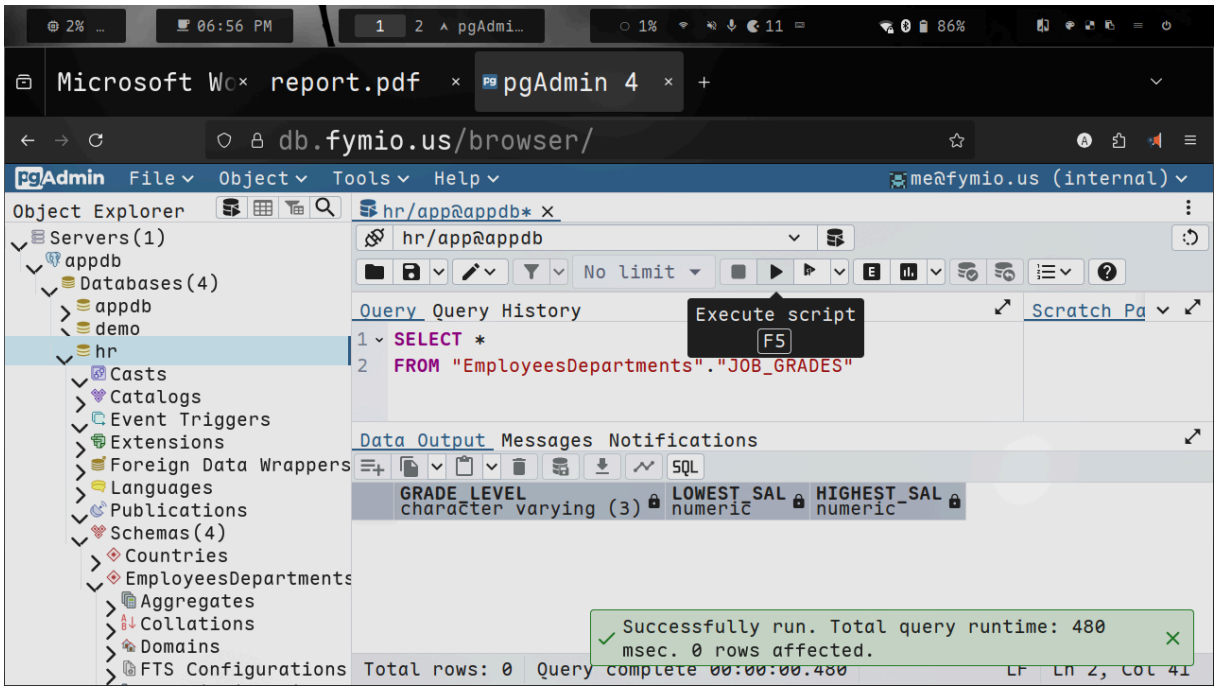

Рис. 1: Исполнение исправленного запроса.

1.2 Команда SELECT содержит 4 ошибки. Укажите их.

```
SELECT "EMPLOYEE_ID", "LAST_NAME"
"SAL" x 12 ANNUAL SALARY
FROM "EmployeesDepartments"."EMPLOYEES"
```

- Нет запятой перед `"SAL"`.
- Использовать `*` вместо `x`.
- Неправильно задан псевдоним: нужен `AS` и, раз в псевдониме пробел, - кавычки (например `AS "ANNUAL SALARY"`).
- Неправильное название стобца: вместо `SAL` должно быть `SALARY`.

Исправленная версия (Рис. 2):

```
SELECT "EMPLOYEE_ID", "LAST_NAME", "SALARY" * 12 AS "ANNUAL SALARY"
FROM "EmployeesDepartments"."EMPLOYEES";
```

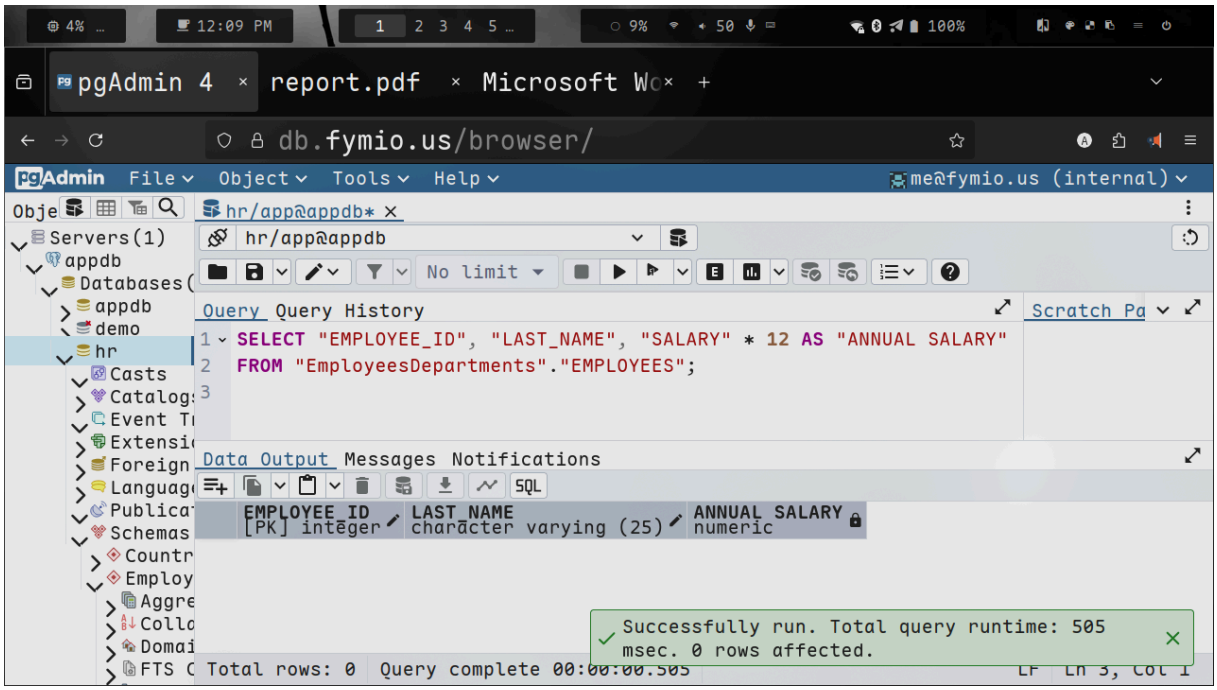

Рис. 2: Исполнение исправленного запроса.

1.3 Напишите запрос, который отображает структуру таблицы `DEPARTMENTS`, представленную на таблице 1. Сформируйте запрос на выборку данных из нее, результат должен соответствовать таблице 2.

column_name	is_nullable	Type
DEPARTMENT_ID	NO	smallint
DEPARTMENT_NAME	NO	character varying
MANAGER_ID	YES	integer
LOCATION_ID	YES	smallint

Табл. 1: Структура таблицы `DEPARTMENTS`.

```
SELECT column_name,
       CASE WHEN is_nullable = 'YES' THEN 'YES' ELSE 'NO' END AS is_nullable,
       data_type
FROM information_schema.columns
WHERE table_name = 'DEPARTMENTS'
  AND table_schema IN ('EmployeesDepartments', 'public')
ORDER BY ordinal_position;
```

Этот запрос извлекает данные таблицы `DEPARTMENTS` из представления `information_schema.columns`.

- `column_name` — имя столбца;
- `is_nullable` — может ли столбец содержать null;
- `data_type` — тип данных столбца.

Фильтрация по `table_name` и `table_schema` позволяет получить только нужную таблицу. Сортировка по `ordinal_position` отображает столбцы в порядке их создания в таблице. Результат выполнения скрипта представлен на рисунке 3.

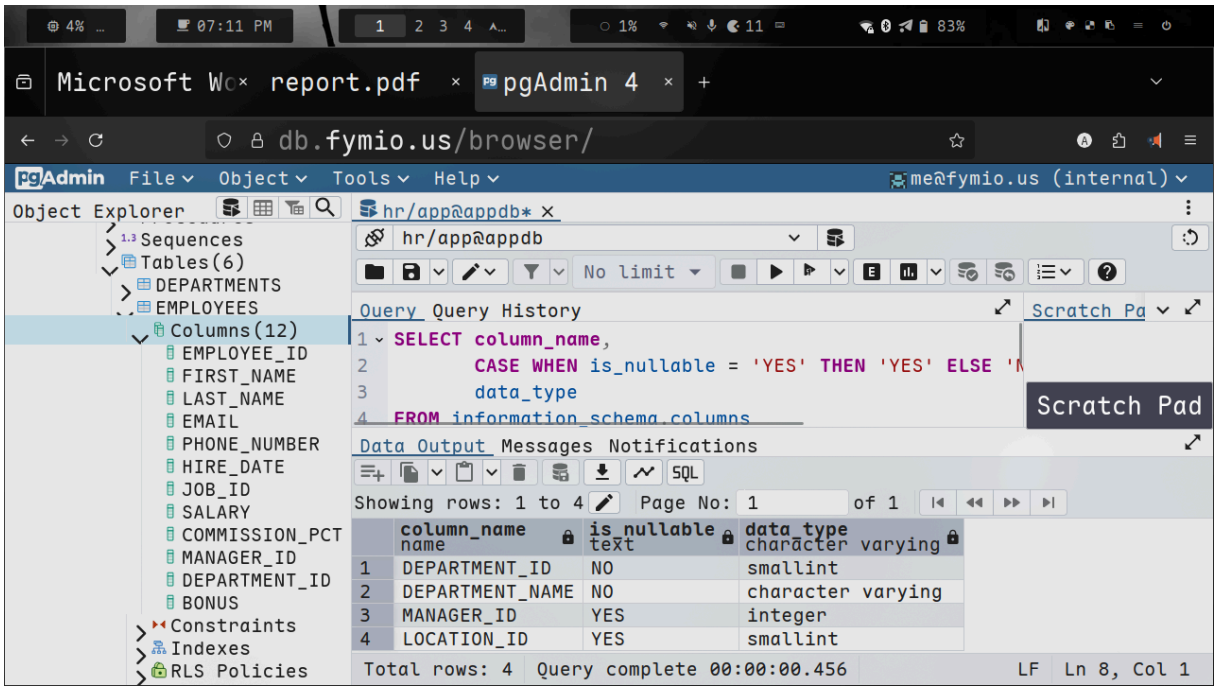

Рис. 3: Результат выполнения запроса.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales - Europe	149	2500
85	Sales - Americas	149	2100
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

Табл. 2: Результат выполнения запроса к таблице `DEPARTMENTS`.

```
SELECT "DEPARTMENT_ID", "DEPARTMENT_NAME", "MANAGER_ID", "LOCATION_ID"
FROM "EmployeesDepartments"."DEPARTMENTS"
ORDER BY "DEPARTMENT_ID";
```

Этот запрос выбирает все данные из таблицы `DEPARTMENTS` в схеме `EmployeesDepartments`. Сортировка по `DEPARTMENT_ID` обеспечивает упорядоченный вывод строк. Результат выполнения запроса представлен на рисунке 4.

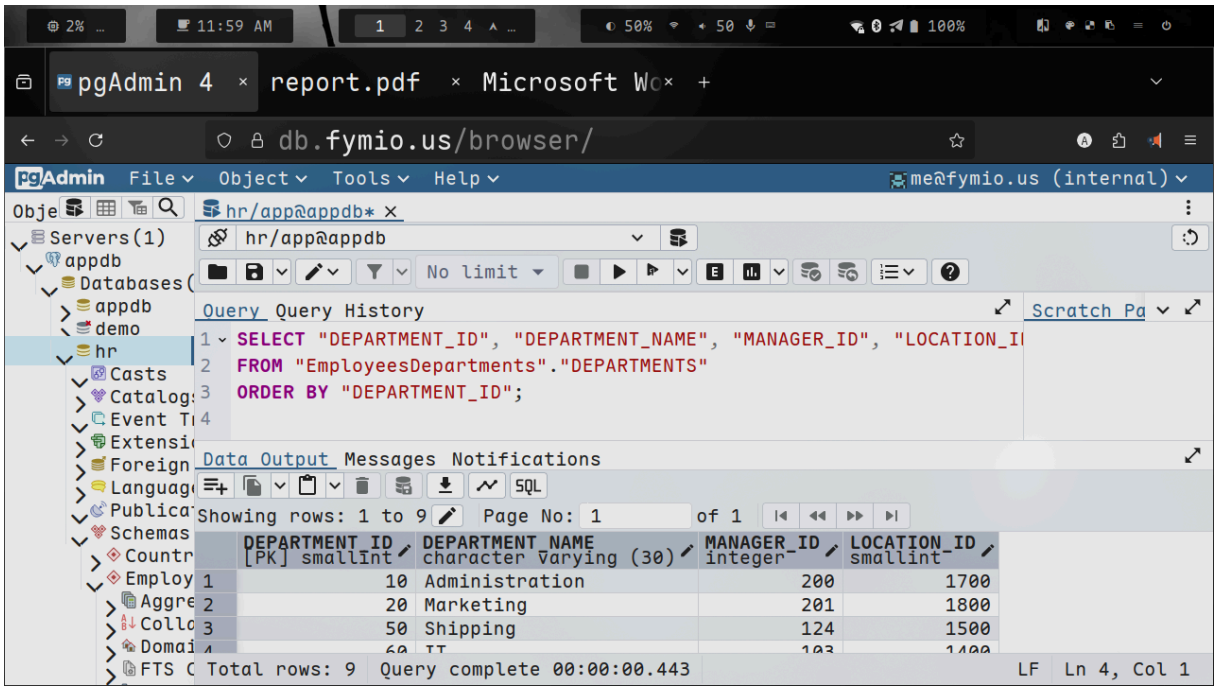

Рис. 4: Результат выполнения запроса.

1.4 Напишите запрос, который отображает структуру таблицы `EMPLOYEES`, представленную в таблице 3.

Name	Null?	Type
HIRE_DATE	NO	date
SALARY	YES	numeric
COMMISION_PCT	YES	numeric
MANAGER_ID	YES	integer
DEPARTMENT_ID	YES	smallint
EMPLOYEE_ID	NO	integer
BONUS	YES	character varying
FIRST_NAME	YES	character varying
LAST_NAME	NO	character varying
EMAIL	NO	character varying
PHONE_NUMBER	YES	character varying
JOB_ID	NO	character varying

Табл. 3: Результат выполнения запроса к таблице `DEPARTMENTS`.

```
SELECT column_name AS "Name",
       CASE WHEN is_nullable = 'YES' THEN 'YES' ELSE 'NO' END AS "Null?",
       data_type AS "Type"
FROM information_schema.columns
WHERE table_schema = 'EmployeesDepartments'
  AND table_name = 'EMPLOYEES'
ORDER BY ordinal_position;
```

Этот запрос извлекает информацию о столбцах таблицы `EMPLOYEES`:

- `Name` — имя столбца;
- `Null?` — может ли столбец содержать NULL;
- `Type` — тип данных столбца.

Фильтрация по `table_schema` и `table_name` позволяет получить данные только для нужной таблицы. Сортировка по `ordinal_position` отображает столбцы в том порядке, в котором они были созданы в таблице. Результат запроса приведен на рисунке 5.

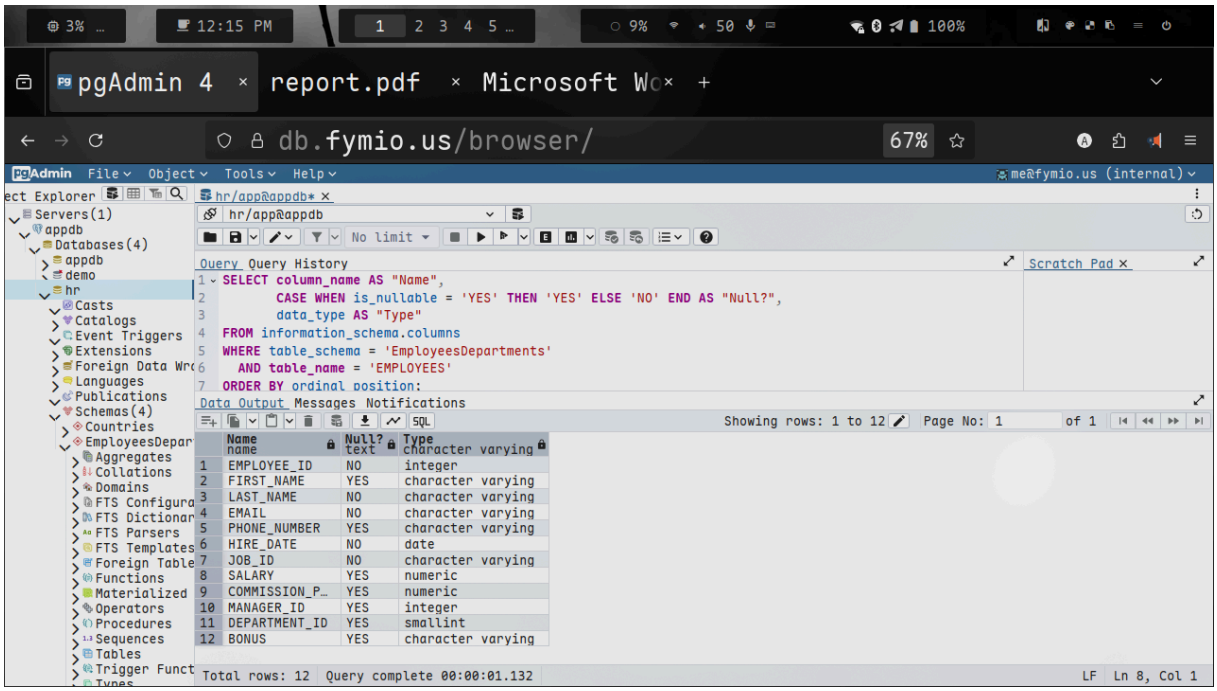

Рис. 5: Результат выполнения запроса.

1.5 Составьте запрос для вывода фамилии каждого служащего, должности, даты найма и номера. Номер служащего должен быть первым. Результат запроса должен быть схож с таблицей 4.

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE
100	King	AD_PRES	2002-06-17
101	Kochhar	AD_VP	2004-09-21
102	De Haan	AD_VP	2008-01-13
103	Hunold	IT_PROG	2005-01-03
104	Ernst	IT_PROG	2006-05-21
107	Lorentz	IT_PROG	2014-02-07
124	Mourgos	ST_MAN	2014-11-16
141	Rajs	ST_CLERK	2010-10-17
142	Davies	ST_CLERK	2012-01-29
143	Matos	ST_CLERK	2013-03-15
144	Vargas	ST_CLERK	2013-07-09
…	…	…	…

Табл. 4: Часть результата выполнения запроса из пункта 1.5.

```
SELECT "EMPLOYEE_ID",
       "LAST_NAME",
       "JOB_ID",
       "HIRE_DATE"
FROM "EmployeesDepartments"."EMPLOYEES"
ORDER BY "EMPLOYEE_ID";
```

В этом запросе выбираются четыре столбца таблицы `EMPLOYEES`:

- `EMPLOYEE_ID` — идентификатор сотрудника, выводится первым;
- `LAST_NAME` — фамилия сотрудника;
- `JOB_ID` — код должности;
- `HIRE_DATE` — дата найма.

Сортировка по `EMPLOYEE_ID` обеспечивает упорядоченный вывод. Использование точных имён столбцов с кавычками гарантирует корректное выполнение запроса в PostgreSQL с регистрозависимыми именами. Результат выполнения запроса представлен на рисунке 6.

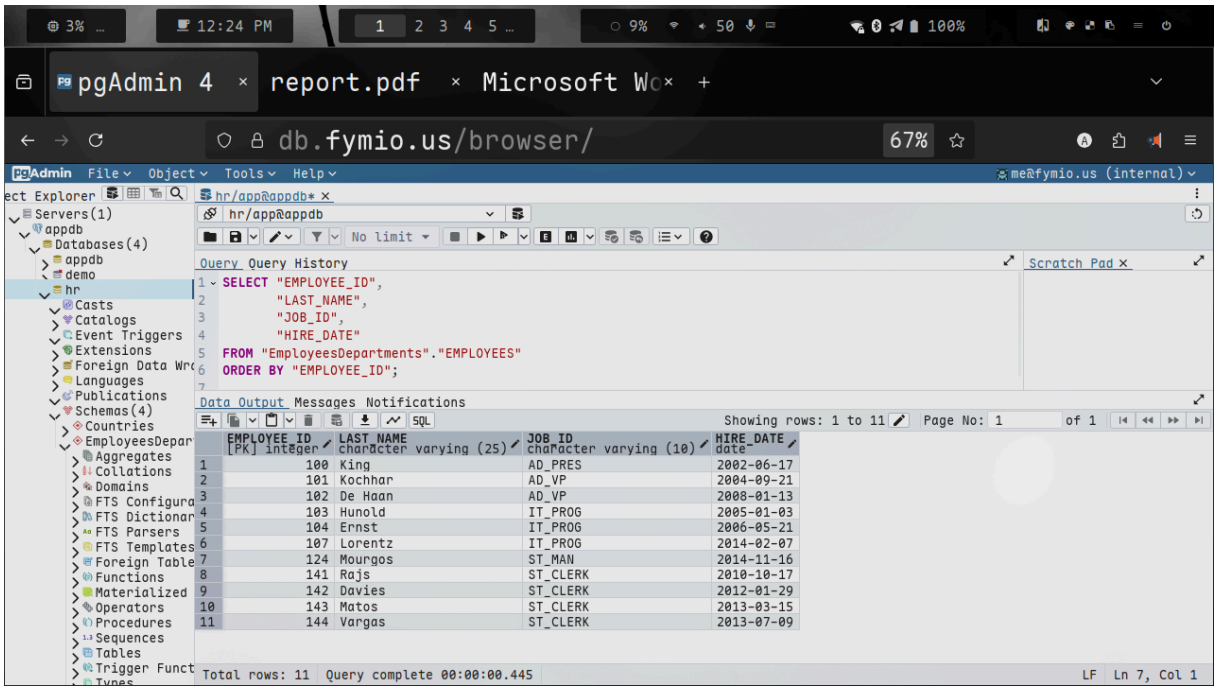

Рис. 6: Результат выполнения запроса.

1.6 Составьте запрос для вывода неповторяющихся должностей из таблицы EMPLOYEES, результат должен соответствовать таблице 5.

JOB_ID
AC_ACCOUNT
AC_MGR
AD_ASST
AD_PRES
AD_VP
IT_PROG
MK_MAN
MK_REP
SA_MAN
SA_REP
SR_MK_REP
SR_SA_REP
SR_ST_CLRK
ST_CLERK
ST_MAN

Табл. 5: Результат выполнения запроса для вывода неповторяющихся должностей.

```
SELECT DISTINCT "JOB_ID"
FROM "EmployeesDepartments"."EMPLOYEES"
ORDER BY "JOB_ID";
```

В этом запросе используется ключевое слово `DISTINCT` для того, чтобы выбрать только уникальные значения столбца `JOB_ID` из таблицы `EMPLOYEES`. Сортировка по `JOB_ID` упорядочивает результат по алфавиту. Результат выполнения запроса представлен на рисунке 7.

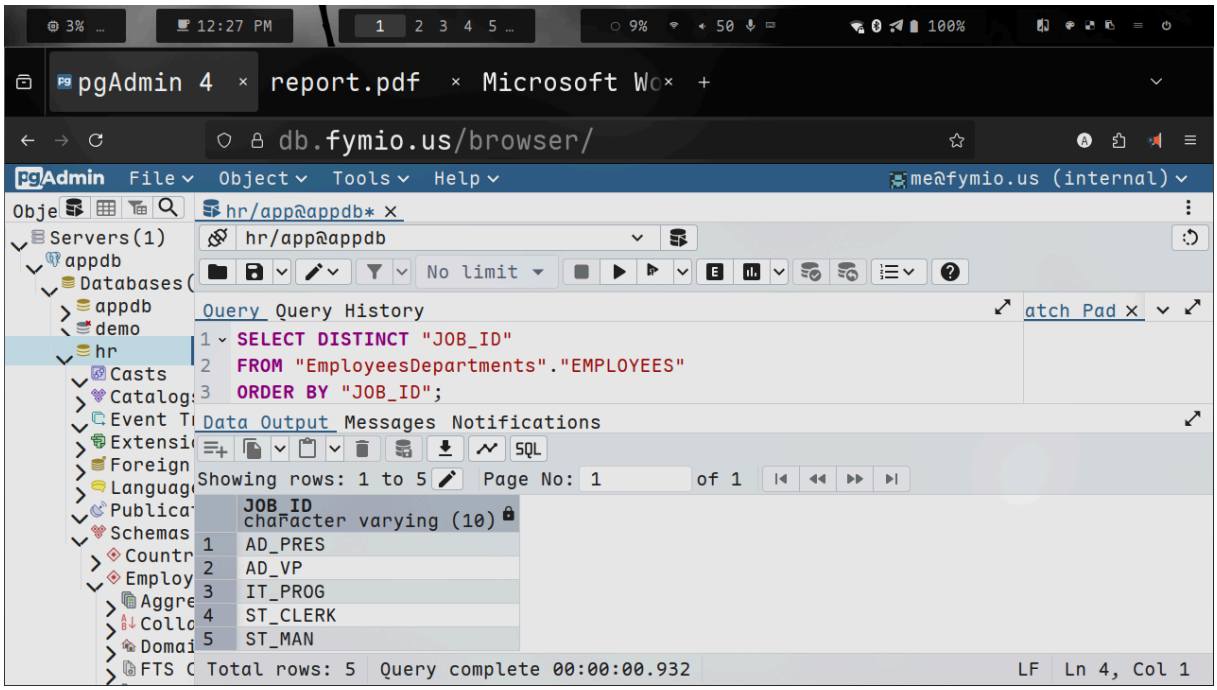

Рис. 7: Результат выполнения запроса.

1.7 Выведите на экран фамилию, соединенную с идентификатором должности через запятую и пробел. Назовите новый столбец Employee and Title. Результат запроса должен быть схож с таблицей 6.

Employee and Title
King, AD_PRES
Kochhar, AD_VP
De Haan, AD_VP
Hunold, IT_PROG
Ernst, IT_PROG
Lorentz, IT_PROG
Mourgos, ST_MAN
Rajs, ST_CLERK
Davies, ST_CLERK
Matos, ST_CLERK
Vargas, ST_CLERK
...

Табл. 6: Результат выполнения запроса для вывода фамилии, соединённой с идентификатором должности.

```
SELECT "LAST_NAME" || ', ' || "JOB_ID" AS "Employee and Title"
FROM "EmployeesDepartments"."EMPLOYEES"
ORDER BY "EMPLOYEE_ID";
```

В этом запросе используется оператор `||` для конкатенации строк:

- сначала `LAST_NAME`,
- затем запятая и пробел `', '`,

- затем `JOB_ID`.

Результат выводится под новым именем столбца `Employee and Title`. Результат выполнения запроса представлен на рисунке 8.

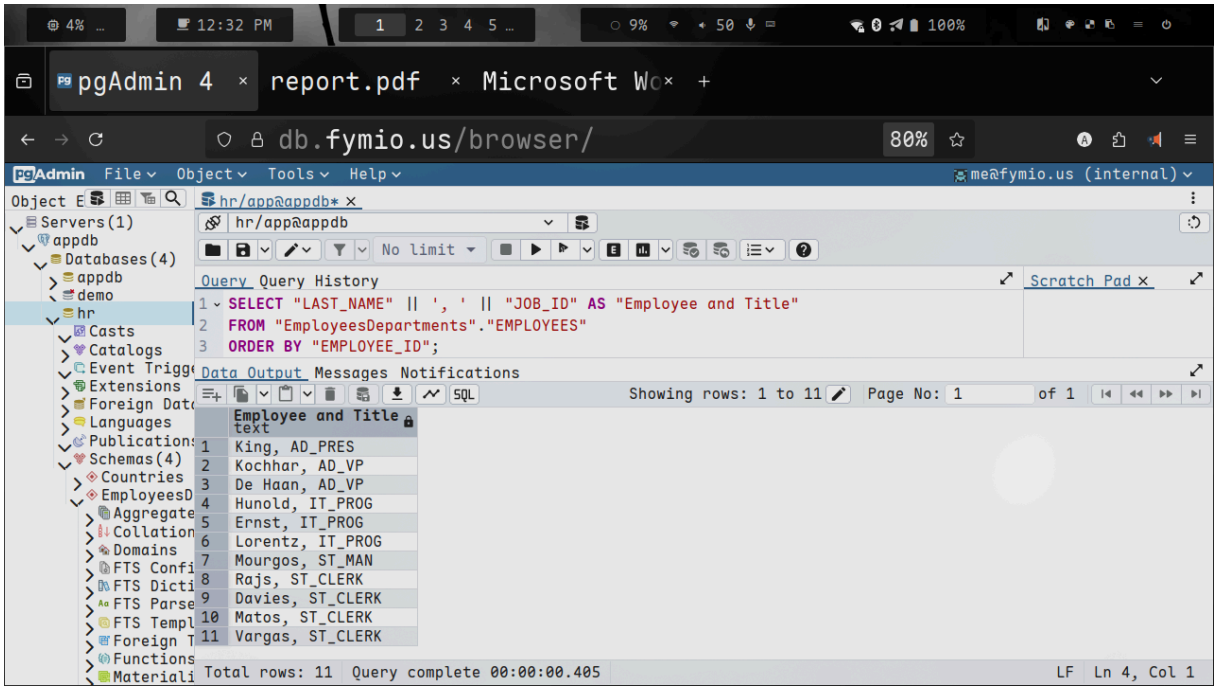

Рис. 8: Результат выполнения запроса.

Задание 2. Выборка данных и изменение последовательности вывода строк, ограничение количества возвращаемых строк с помощью предложения `WHERE`, сортировка строк с помощью предложения `ORDER BY`.

2.1 Создайте запрос для вывода фамилии и заработной платы служащих, зарабатывающих более 12000. Результат выполнения запроса должен соответствовать таблице 7.

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hartstein	13000

Рис. 7: Результат выполнения запроса для вывода фамилии и заработной платы служащих

```
SELECT "LAST_NAME", "SALARY"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "SALARY" > 12000
ORDER BY "SALARY" DESC;
```

- выбираются столбцы `LAST_NAME` и `SALARY` из таблицы `EMPLOYEES`;
- фильтруются только те строки, где заработная плата больше 12000;
- строки сортируются по убыванию зарплаты, чтобы сначала отображались сотрудники с наибольшей зарплатой.

Результат выполнения запроса представлен на рисунке 9.

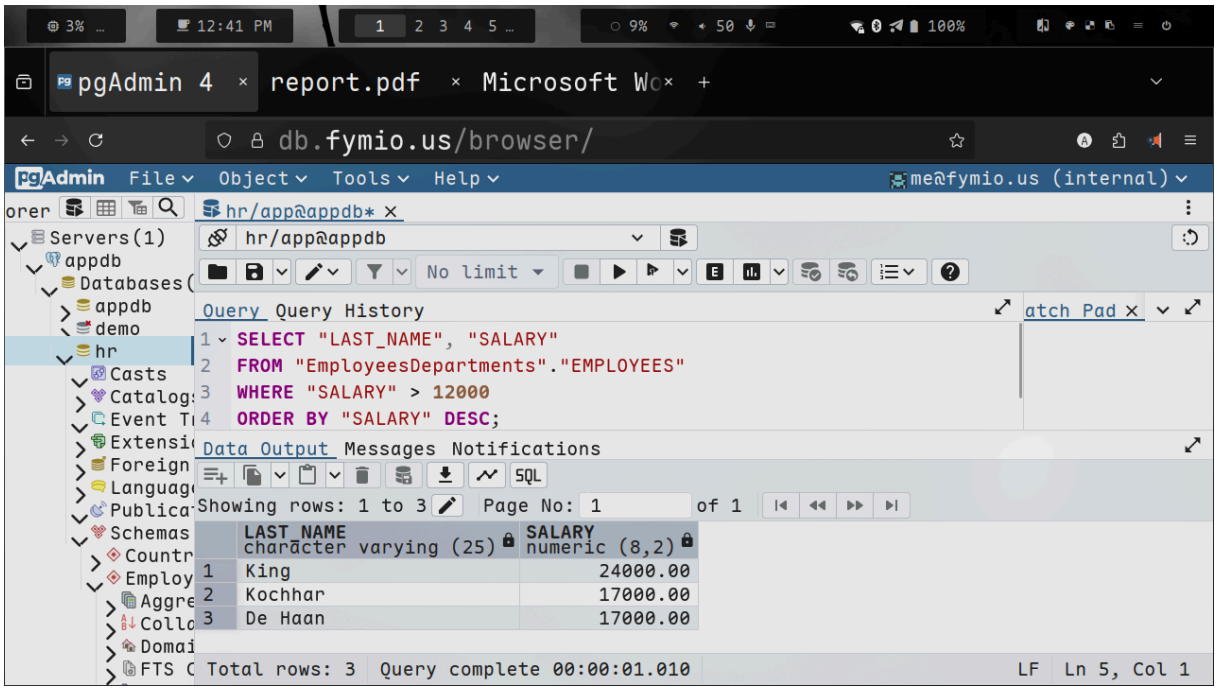

Рис. 9: Результат выполнения запроса.

2.2 Создайте запрос для вывода фамилии и номера отдела служащего под номером 176. Результат выполнения запроса должен соответствовать таблице 7.

LAST_NAME	DEPARTMENT_ID
Taylor	80

Табл. 8: Результат выполнения запроса для служащего под номером 176.

```
SELECT "LAST_NAME", "DEPARTMENT_ID"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "EMPLOYEE_ID" = 176;
```

- выбираются столбцы `LAST_NAME` и `DEPARTMENT_ID` из таблицы `EMPLOYEES`;
- фильтрация выполняется по точному значению `EMPLOYEE_ID` с помощью условия `WHERE`;

Результат выполнения запроса представлен на рисунке 10.

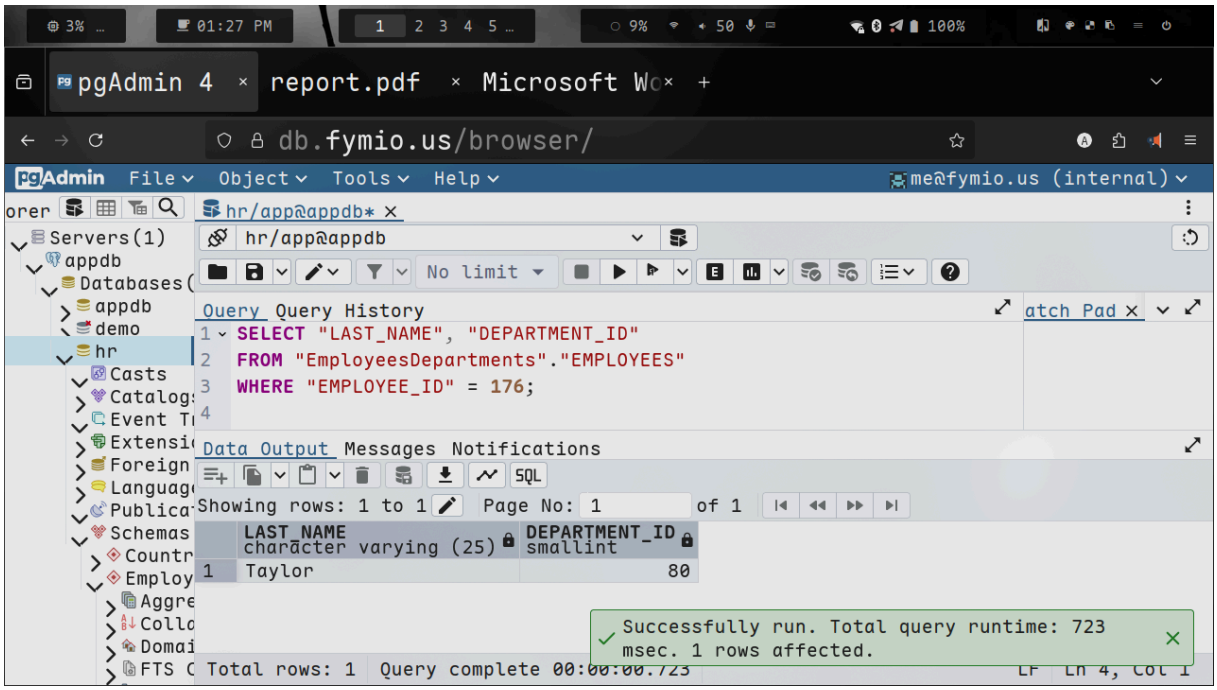

Рис. 10: Результат выполнения запроса.

2.3 Измените запрос из задания 2.1 и выведите фамилии и оклады всех служащих, чей оклад не входит в диапазон от 5000 до 12000. Результат выполнения запроса должен быть схож с таблицей 8.

LAST_NAME	SALARY
King	24000.00
Kochhar	17000.00
De Haan	17000.00
Lorentz	4200.00
Rajs	3500.00
Davies	3100.00
Matos	2600.00
Vargas	2500.00
Whalen	4400.00
Hartstein	13000.00

Табл. 9: Результат выполнения запроса для служащих, чей оклад не входит в диапазон от 5000 до 12000.

```
SELECT "LAST_NAME", "SALARY"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "SALARY" < 5000 OR "SALARY" > 12000
ORDER BY "SALARY" DESC;
```

- выбираются столбцы `LAST_NAME` и `SALARY` из таблицы `EMPLOYEES`;
- фильтруются строки, где оклад не входит в диапазон от 5000 до 12000, с помощью условия `WHERE ... OR`;
- строки сортируются по убыванию оклада.

Результат выполнения запроса представлен на рисунке 11.

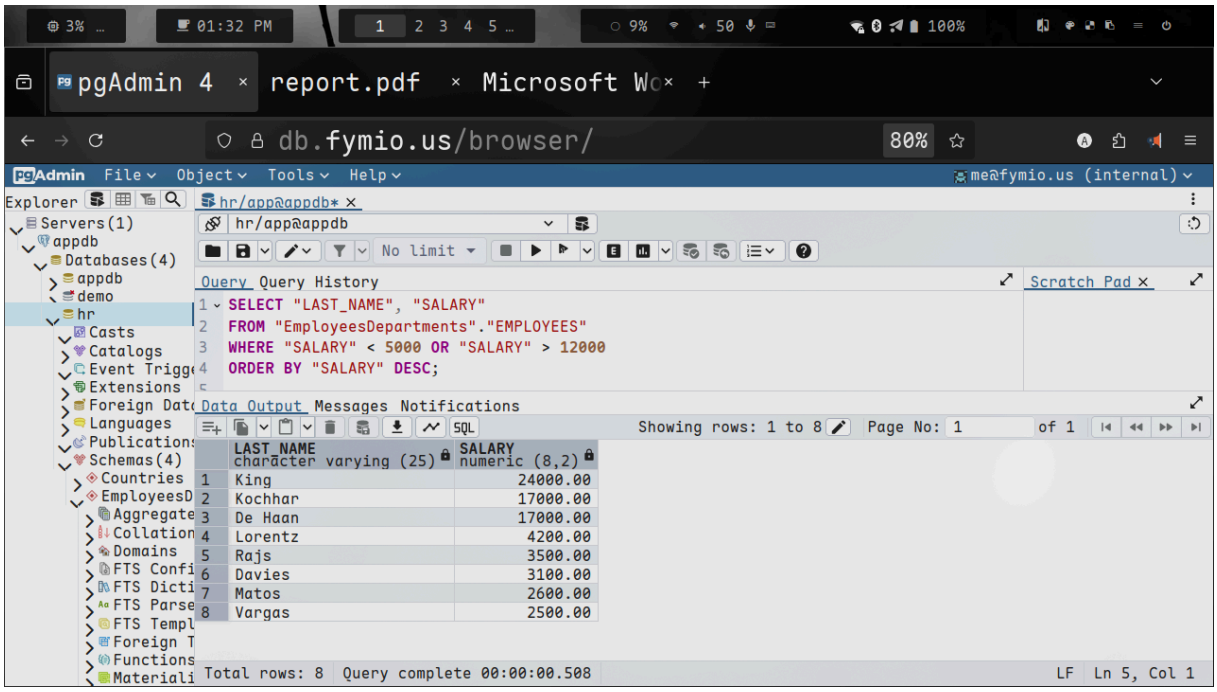

Рис. 11: Результат выполнения запроса.

2.4 Выведите фамилию, идентификатор должности и дату начала работы всех служащих, нанятых в период с 16 февраля 2011 по 12 мая 2011г. Отсортируйте данные в порядке возрастания даты найма. Результат выполнения запроса должен соответствовать таблице 9.

LAST_NAME	JOB_ID	HIRE_DATE
Hartstein	MK_MAN	2011-02-17
Abel	SA_REP	2011-05-11

Табл. 10: Результат выполнения запроса для служащих, нанятых в период с 16 февраля 2011 по 12 мая 2011г.

```
SELECT "LAST_NAME", "JOB_ID", "HIRE_DATE"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "HIRE_DATE" BETWEEN '2011-02-16' AND '2011-05-12'
ORDER BY "HIRE_DATE" ASC;
```

- выбираются столбцы `LAST_NAME`, `JOB_ID` и `HIRE_DATE` из таблицы `EMPLOYEES`;
- фильтрация выполняется по дате найма с помощью предложения `WHERE` и `BETWEEN`, чтобы выбрать сотрудников, нанятых с 16 февраля по 12 мая 2011 года включительно;
- сортировка по возрастанию даты найма.

Результат выполнения запроса представлен на рисунке 12.

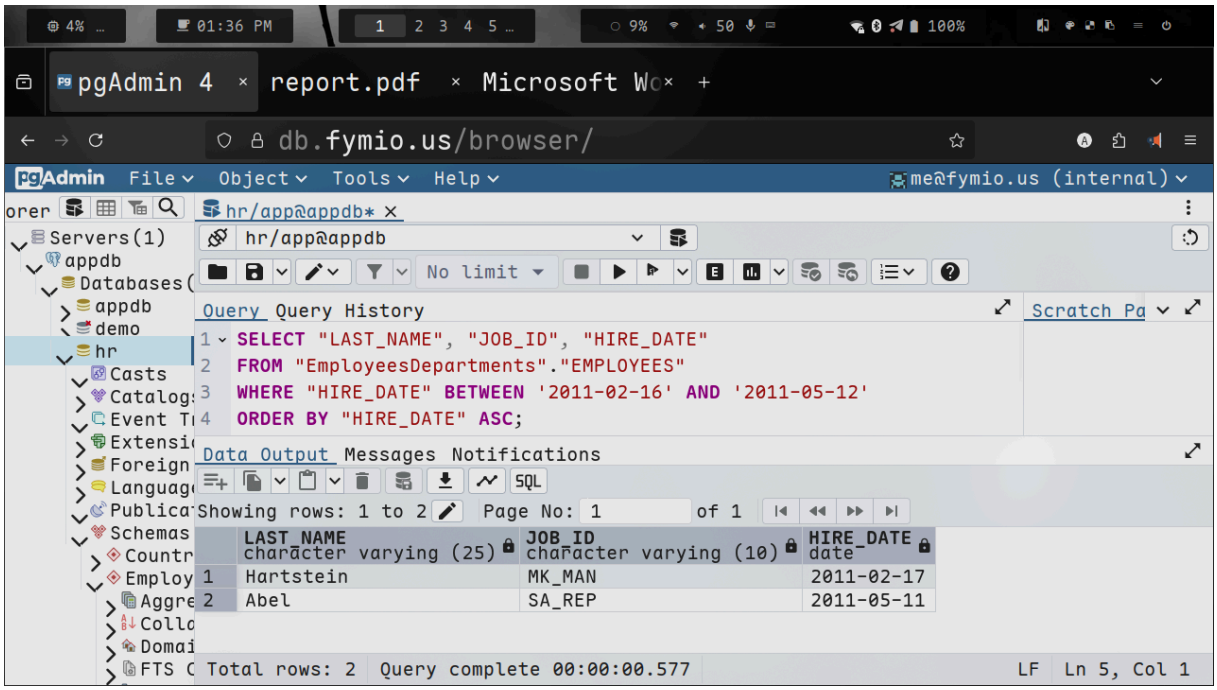

Рис. 12: Результат выполнения запроса.

2.5 Выведите фамилию и номер отдела всех служащих из отделов 20 и 50. Отсортируйте данные по фамилиям в алфавитном порядке. Результат выполнения запроса должен быть схож с таблицей 10.

LAST_NAME	DEPARTMENT_ID
Bell	50
Davies	50
Fay	20
Hartstein	20
Heiden	50
Matos	50
Mourgos	50

Табл. 11: Результат выполнения запроса для служащих из отделов 20 и 50.

```
SELECT "LAST_NAME", "DEPARTMENT_ID"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "DEPARTMENT_ID" IN (20, 50)
ORDER BY "LAST_NAME" ASC;
```

- выбираются столбцы `LAST_NAME` и `DEPARTMENT_ID` из таблицы `EMPLOYEES`;
- фильтруются сотрудники, работающие в отделах 20 и 50 с помощью условия `WHERE ... IN`;
- сортировка по фамилии.

Результат выполнения запроса представлен на рисунке 13.

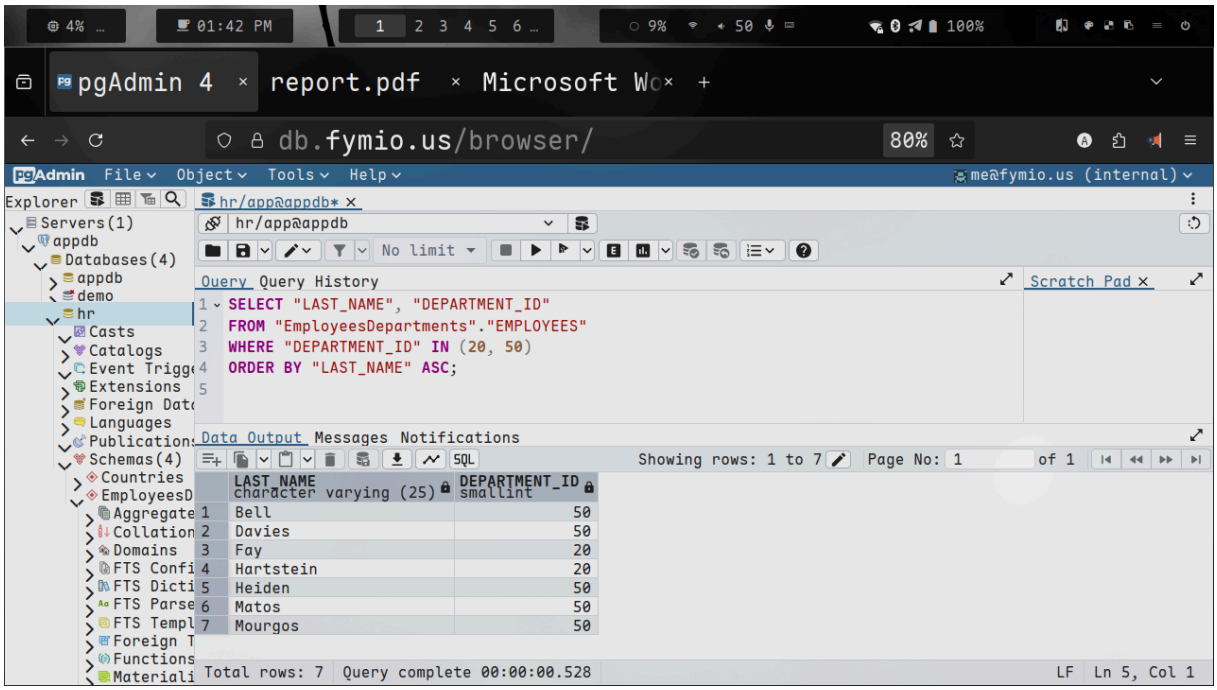

Рис. 13: Результат выполнения запроса.

2.6 Измените запрос из задания 2.3 для вывода фамилий и окладов служащих отделов 20 и 50, зарабатывающих от 5000 до 12000. Назовите столбцы Employee и Mounthly Salary, соответственно. Результат выполнения запроса должен соответствовать таблице 11.

EMPLOYEE	Mounthly Salary
Safwah	5000.00
Mourgos	5800.00
Steiner	8600.00

Табл. 12: Результат выполнения запроса для служащих из отделов 20 и 50, зарабатывающих от 5000 до 12000.

```
SELECT "LAST_NAME" AS "EMPLOYEE", "SALARY" AS "Mounthly Salary"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "DEPARTMENT_ID" IN (20, 50)
  AND "SALARY" BETWEEN 5000 AND 12000
ORDER BY "LAST_NAME" ASC;
```

- выбираются столбцы `LAST_NAME` и `SALARY` из таблицы `EMPLOYEES`;
- столбцы переименовываются через `AS` в `EMPLOYEE` и `Mounthly Salary`;
- фильтруются сотрудники, работающие в отделах 20 и 50 с зарплатой от 5000 до 12000 с помощью `WHERE ... AND ... BETWEEN`;
- сортировка по фамилии.

Результат выполнения запроса представлен на рисунке 14.

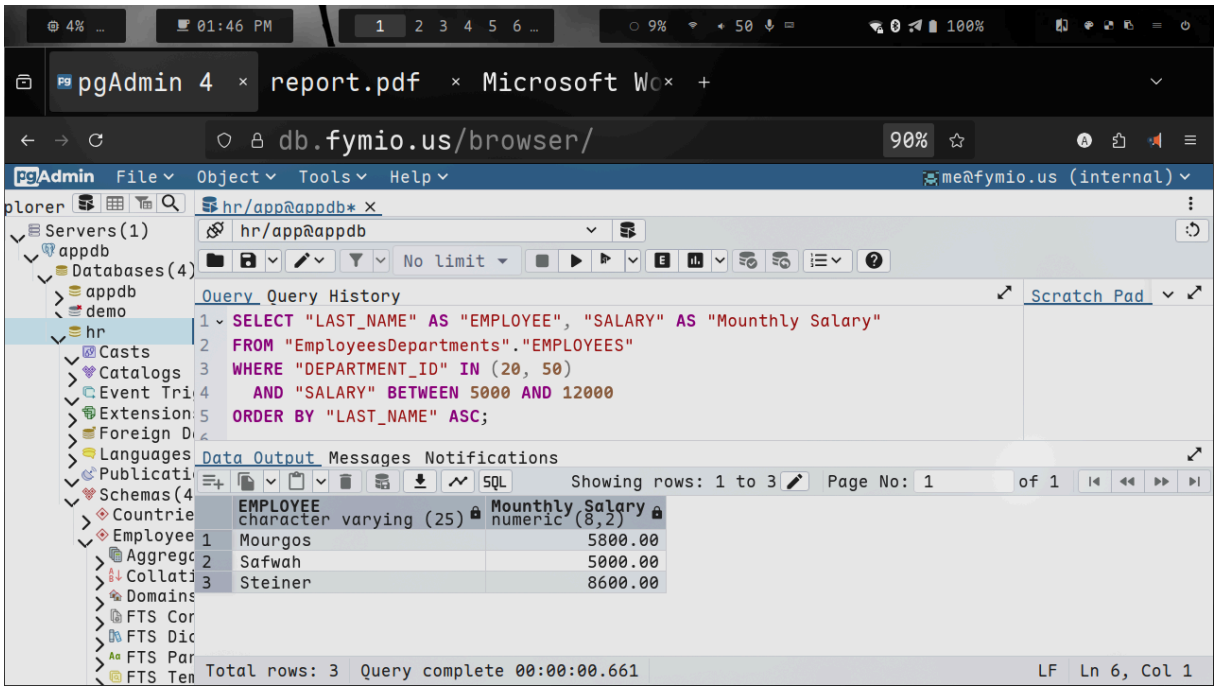

Рис. 14: Результат выполнения запроса.

2.7 Выведите фамилии и должности всех служащих, не имеющих менеджера. Результат выполнения запроса должен соответствовать таблице 12.

LAST_NAME	JOB_ID
King	AD_PRES

Табл. 13: Результат выполнения запроса для служащих, не имеющих менеджера.

```
SELECT "LAST_NAME", "JOB_ID"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "MANAGER_ID" IS NULL;
```

- выбираются столбцы `LAST_NAME` и `JOB_ID` из таблицы `EMPLOYEES`;
- фильтруются сотрудники, у которых нет менеджера, с помощью условия `WHERE "MANAGER_ID" IS NULL`;
- результат содержит только тех сотрудников, которые не подчиняются никому.

Результат выполнения запроса представлен на рисунке 15.

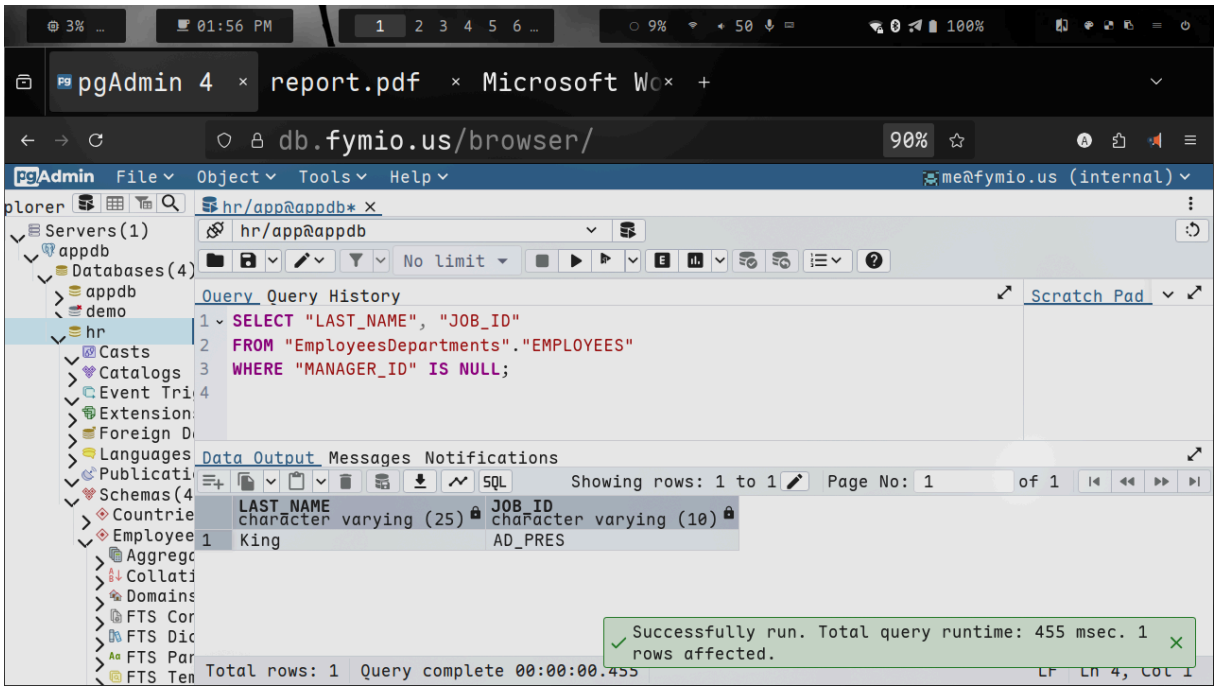

Рис. 15: Результат выполнения запроса.

2.8 Выведите фамилию, и комиссионные всех служащих, зарабатывающих комиссионные. Отсортируйте данные в порядке убывания окладов и комиссионных. Результат выполнения запроса должен соответствовать таблице 13.

LAST_NAME	SALARY	COMMISION_PCT
Grant	7000.00	0.15
Alves Rocha	7300.00	0.15
Almeida Castro	7300.00	0.20
Silva Pinto	7500.00	0.15
Taylor	8600.00	0.20
Barbosa Souza	9500.00	0.20
Hooper	9600.00	0.20
Zlotkey	10500.00	0.20
Abel	11000.00	0.30

Табл. 14: Результат выполнения запроса для служащих, зарабатывающих комиссионные.

```
SELECT "LAST_NAME", "SALARY", "COMMISSION_PCT"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "COMMISSION_PCT" IS NOT NULL
ORDER BY "SALARY" DESC, "COMMISSION_PCT" DESC;
```

- `WHERE "COMMISSION_PCT" IS NOT NULL` — отбирает только сотрудников, которые получают комиссионные.
- `ORDER BY "SALARY" DESC, "COMMISSION_PCT" DESC` — сортировка по зарплате сначала от большей к меньшей, а при равной зарплате — по комиссионным.

Результат выполнения запроса представлен на рисунке 16.

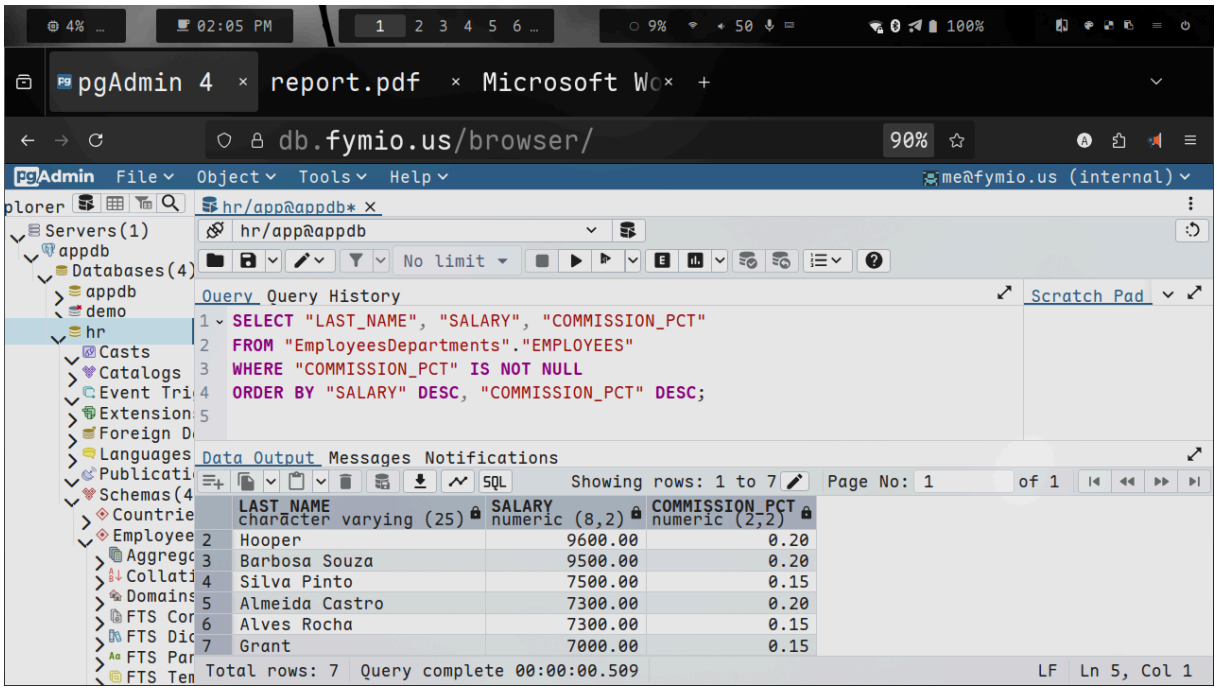

Рис. 16: Результат выполнения запроса.

2.9 Выведите все фамилии служащих БЕЗ использования строковых функций, в которых третья буква - *a*. Результат выполнения запроса должен соответствовать таблице 14.

LAST_NAME
Grant
Whalen

Табл. 15: Результат выполнения запроса (фамилии служащих, в которых третья буква *a*).

```
SELECT "LAST_NAME"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "LAST_NAME" LIKE '__a%'
ORDER BY "LAST_NAME" ASC;
```

- используется оператор `LIKE` для поиска шаблона в столбце `LAST_NAME`;
- `'__a%'` означает:
  - первые две символа могут быть любыми,
  - третий символ — буква ‘a’,
  - `%` соответствует любому количеству оставшихся символов;
- сортировка по фамилии.

Результат выполнения запроса представлен на рисунке 17.

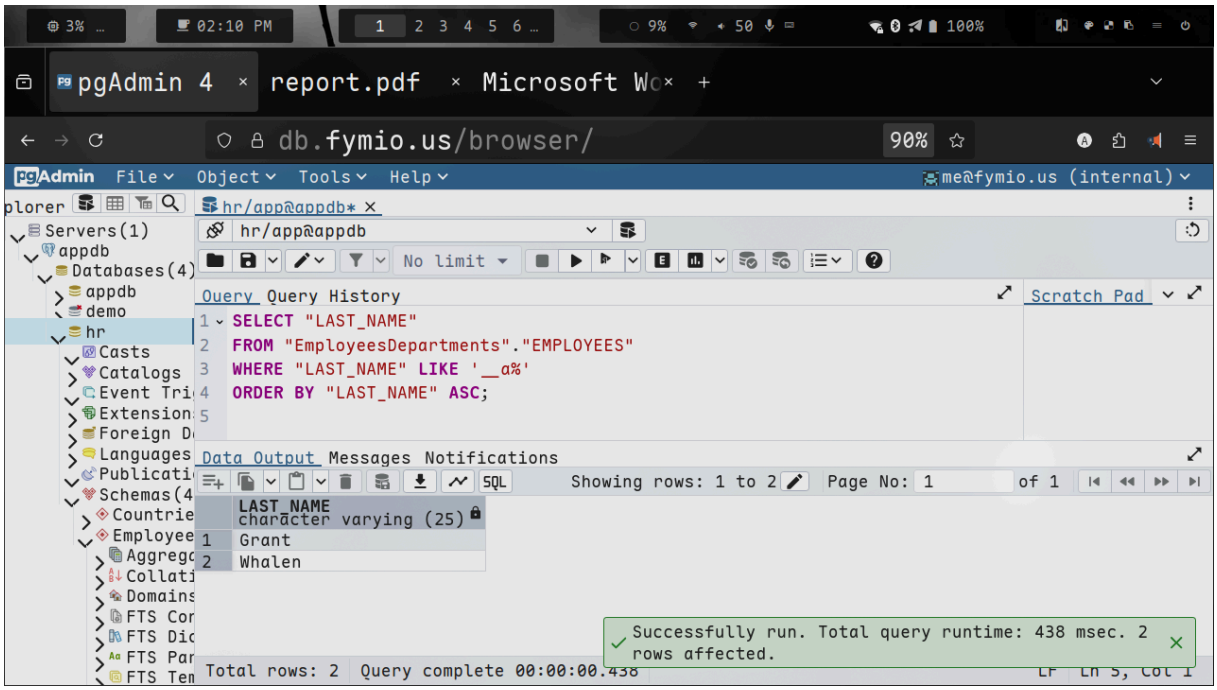

Рис. 17: Результат выполнения запроса.

2.10 Выведите фамилии, должности и оклады всех служащих, работающих торговыми представителями (SA_REP) или клерками на складе (ST_CLERK), с окладом не равным 2600, 3100, 8600 и 11000. Результат выполнения запроса должен соответствовать таблице 15.

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500.00
Vargas	ST_CLERK	2500.00
Grant	SA_REP	7000.0
Silva Pinto	SA_REP	7500.00
Alves Rocha	SA_REP	7300.00
Almeida Castro	SA_REP	7300.00

Табл. 16: Результат выполнения запроса.

```
SELECT "LAST_NAME", "JOB_ID", "SALARY"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "JOB_ID" IN ('SA_REP', 'ST_CLERK')
  AND "SALARY" NOT IN (2600, 3100, 8600, 11000)
ORDER BY "LAST_NAME" ASC;
```

- выбираются столбцы `LAST_NAME`, `JOB_ID` и `SALARY` из таблицы `EMPLOYEES`;
- фильтруются только сотрудники с должностями `SA_REP` или `ST_CLERK` с помощью условия `WHERE ... IN`;
- дополнительно исключаются строки с окладами 2600, 3100, 8600 и 11000 с помощью `NOT IN`;
- сортировка по фамилии.

Результат выполнения запроса представлен на рисунке 18.

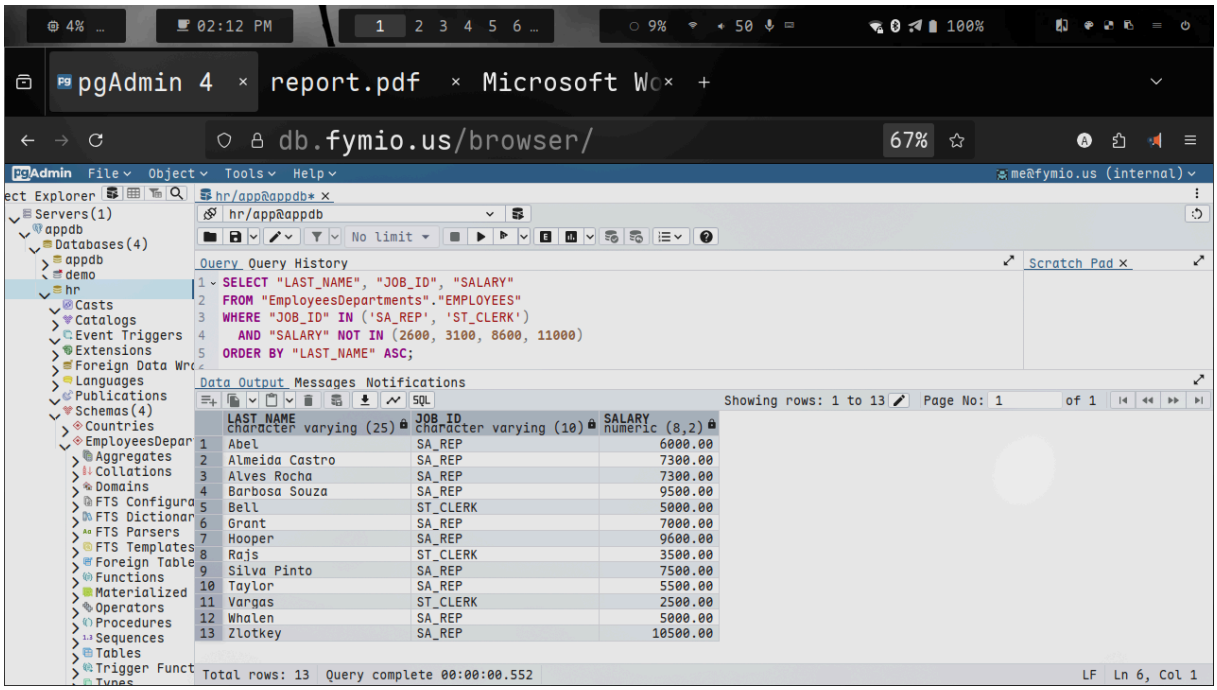

Рис. 18: Результат выполнения запроса.

2.11 Измените запрос 2.8 так, чтобы получить фамилии, оклады и комиссионные всех служащих, у которых сумма комиссионных равна или превышает 20%. Выполните запрос еще раз. Результат выполнения запроса должен соответствовать таблице 16.

Employee	Monthly Salary	COMMISION_PCT
Almeida Castro	7300.00	0.20
Taylor	8600.00	0.20
Barbosa Souza	9500.00	0.20
Hooper	9600.00	0.20
Zlotkey	10500.00	0.20
Abel	11000.00	0.30

Табл. 17: Результат выполнения запроса.

```
SELECT "LAST_NAME" AS "Employee",
       "SALARY" AS "Monthly Salary",
       "COMMISSION_PCT"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE "COMMISSION_PCT" >= 0.20
ORDER BY "LAST_NAME" ASC;
```

- выбираются фамилии сотрудников `LAST_NAME`, месячная зарплата `SALARY` и процент комиссионных `COMMISSION_PCT`;
- фильтруются только те сотрудники, у которых `COMMISSION_PCT` ≥ 0.20, то есть сумма комиссионных равна или превышает 20%;
- столбцы переименованы с помощью `AS`;
- сортировка по фамилии.

Результат выполнения запроса представлен на рисунке 19.

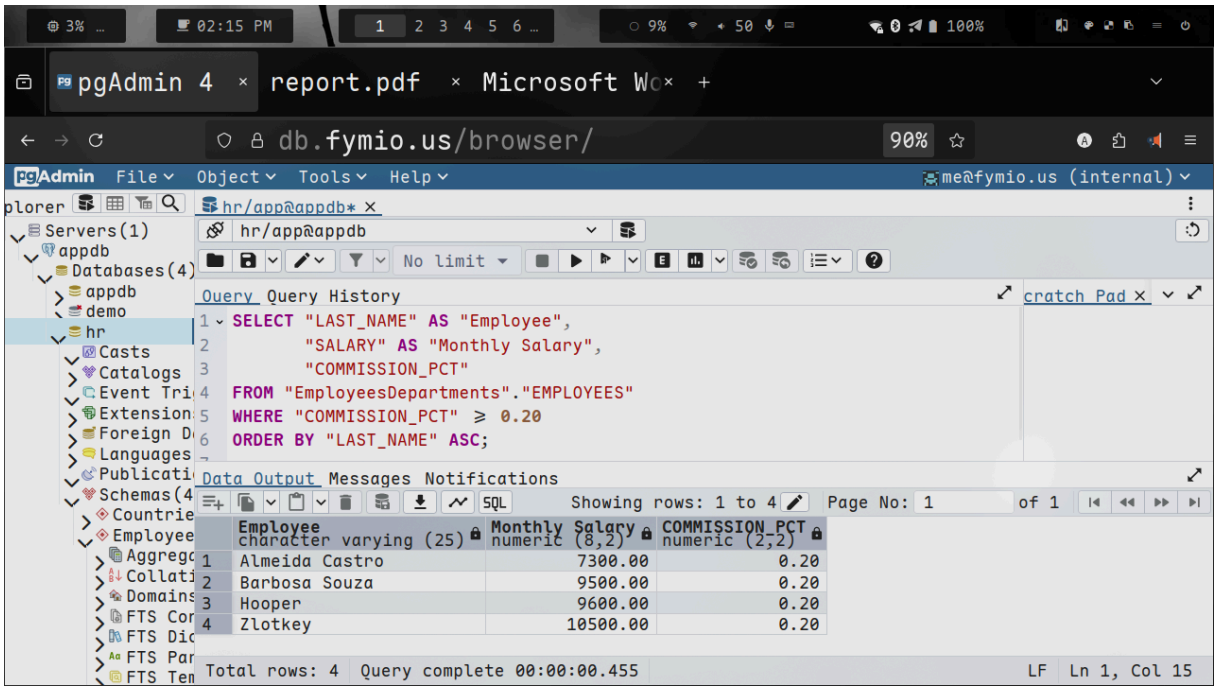

Рис. 19: Результат выполнения запроса.

Задание 3. Составление запросов, требующих использования числовых функций (TRUNC, ROUND и т.д.)

3.1 Выведите номер служащего, его фамилию, оклад, повышенный на 9.5%. Новый оклад должен быть округлен до целого. Назовите столбец New Salary. Результат выполнения запроса должен быть подобен результату, представленному в таблице 17.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
100	King	24000.00	26280
101	Kochhar	17000.00	18615
102	De Haan	17000.00	18615
103	Hunold	9000.00	9855
104	Ernst	6000.00	6570
…	…	…	…

Табл. 18: Результат выполнения запроса.

```
SELECT "EMPLOYEE_ID",
       "LAST_NAME",
       "SALARY",
       ROUND("SALARY" * 1.095) AS "New Salary"
FROM "EmployeesDepartments"."EMPLOYEES"
ORDER BY "EMPLOYEE_ID";
```

- выбираются столбцы `EMPLOYEE_ID`, `LAST_NAME` и `SALARY` из таблицы `EMPLOYEES`;
- новый столбец `New Salary` рассчитывается как оклад, увеличенный на 9.5%: `SALARY` · 1.095;
- функция `ROUND` округляет полученное значение до целого числа;
- с помощью `ORDER BY "EMPLOYEE_ID"` строки сортируются по номеру сотрудника.

Результат выполнения запроса представлен на рисунке 20.

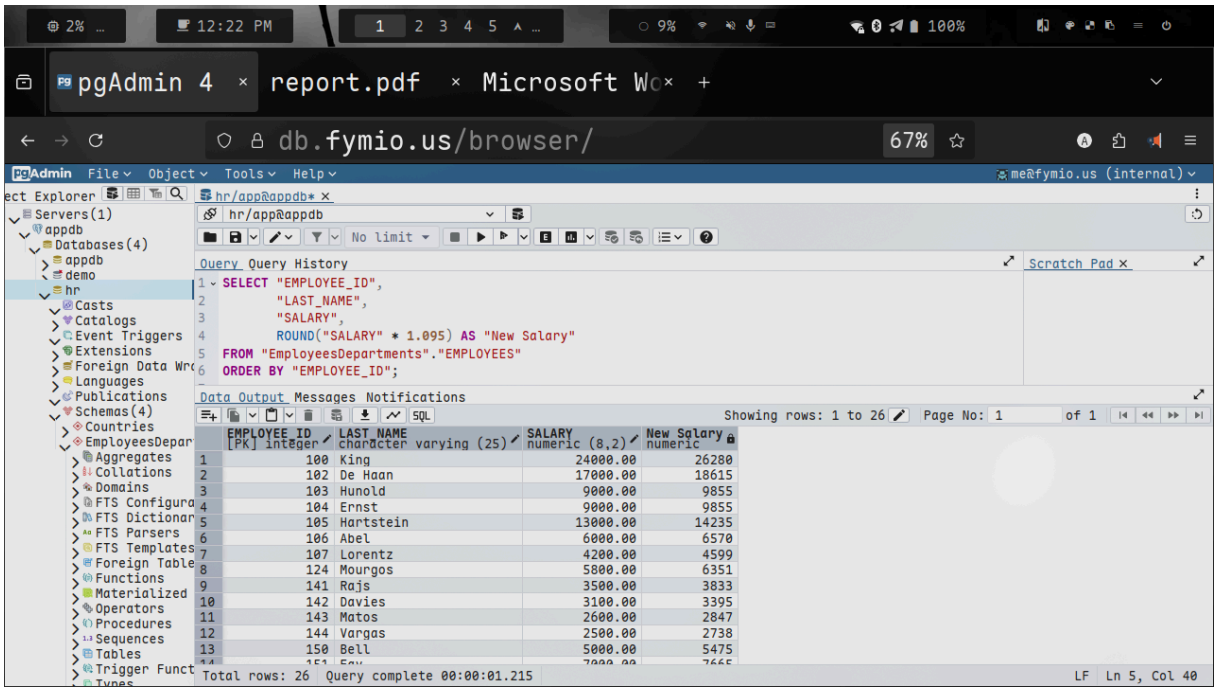

Рис. 20: Результат выполнения запроса.

3.2 Измените запрос из пункта 3.1, добавив еще один столбец, который будет содержать результат вычитания старого оклада из нового. Назовите столбец Increase. Результат выполнения запроса должен быть подобен результату, представленному в таблице 18.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
100	King	24000.00	26280	2280
101	Kochhar	17000.00	18615	1615
102	De Haan	17000.00	18615	1615
103	Hunold	9000.00	9855	855
104	Ernst	6000.00	6570	570
…	…	…	…	…

Табл. 19: Результат выполнения запроса.

```
SELECT "EMPLOYEE_ID",
       "LAST_NAME",
       "SALARY",
       ROUND("SALARY" * 1.095) AS "New Salary",
       ROUND("SALARY" * 1.095) - "SALARY" AS "Increase"
FROM "EmployeesDepartments"."EMPLOYEES"
ORDER BY "EMPLOYEE_ID";
```

- используется выражение из задания 3.1 для расчёта нового оклада, округлённого функцией `ROUND`;
- добавлен новый столбец `Increase`, который вычисляется как разница между новым и старым окладом;

- это позволяет увидеть, насколько увеличилась зарплата каждого сотрудника после повышения на 9.5%;
- строки сортируются по идентификатору сотрудника.

Результат выполнения запроса представлен на рисунке 21.

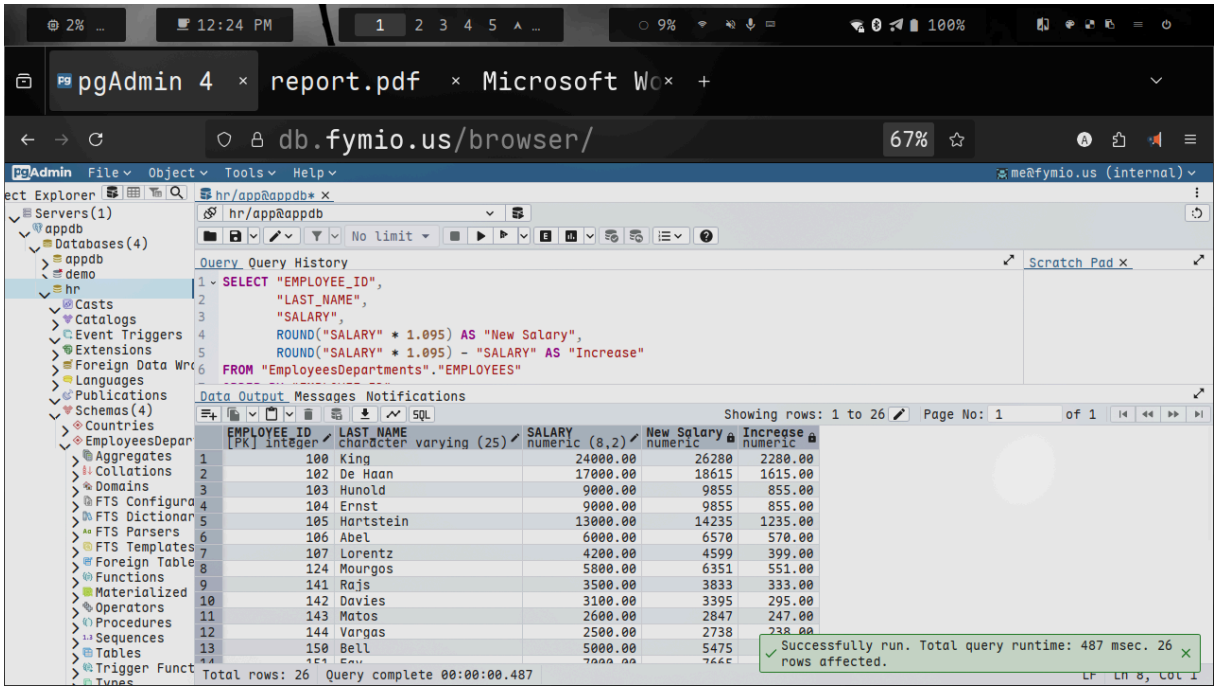

Рис. 21: Результат выполнения запроса.

Задание 4. Составление запросов, требующих символьных функций (INITCAP, POSITION, LENGTH, CONCAT и т.д.)

4.1 Создайте запрос для вывода всех данных из таблицы EMPLOYEES. Разделите столбцы запятыми. Назовите столбец THE_OUTPUT. Результат запроса должен быть схож с таблицей 19.

THE_OUTPUT
100, Steven, King, SKING, 515.123.4567, 2002-06-17, AD_PRES, 24000.00, , , 90
101, Neena, Kochhar, NKOCHHAR, 515.123.4568, 2004-09-21, AD_VP, 17000.00, , 100, 90
102, Lex, De Haan, LDEHAAN, 515.123.4569, 2008-01-13, AD_VP, 17000.00, , 100, 90
103, Alexander, Hunold, AHUNOLD, 590.423.4567, 2005-01-03, IT_PROG, 9000.00, , 102, 60
104, Bruce, Ernst, BERNST, 590.423.4568, 2006-05-21, IT_PROG, 6000.00, , 103, 60
107, Diana, Lorentz, DLORENTZ, 590.423.5567, 2014-02-07, IT_PROG, 4200.00, , 103, 60
…

Табл. 20: Результат выполнения запроса.

```
SELECT
  CONCAT(
    "EMPLOYEE_ID", ', ',
    "FIRST_NAME", ', ',
    "LAST_NAME", ', ',
    "EMAIL", ', ',
```

```
    "PHONE_NUMBER", ', ',
    "HIRE_DATE", ', ',
    "JOB_ID", ', ',
    "SALARY", ', ',
    "COMMISSION_PCT", ', ',
    "MANAGER_ID", ', ',
    "DEPARTMENT_ID"
  ) AS "THE_OUTPUT"
FROM "EmployeesDepartments"."EMPLOYEES";
```

- каждое значение соединяется через запятую и пробел ‘, ‘;
- функция `CONCAT` объединяет текстовые фрагменты;
- все данные из таблицы выводятся в один столбец `THE_OUTPUT`.

Результат выполнения запроса представлен на рисунке 22.

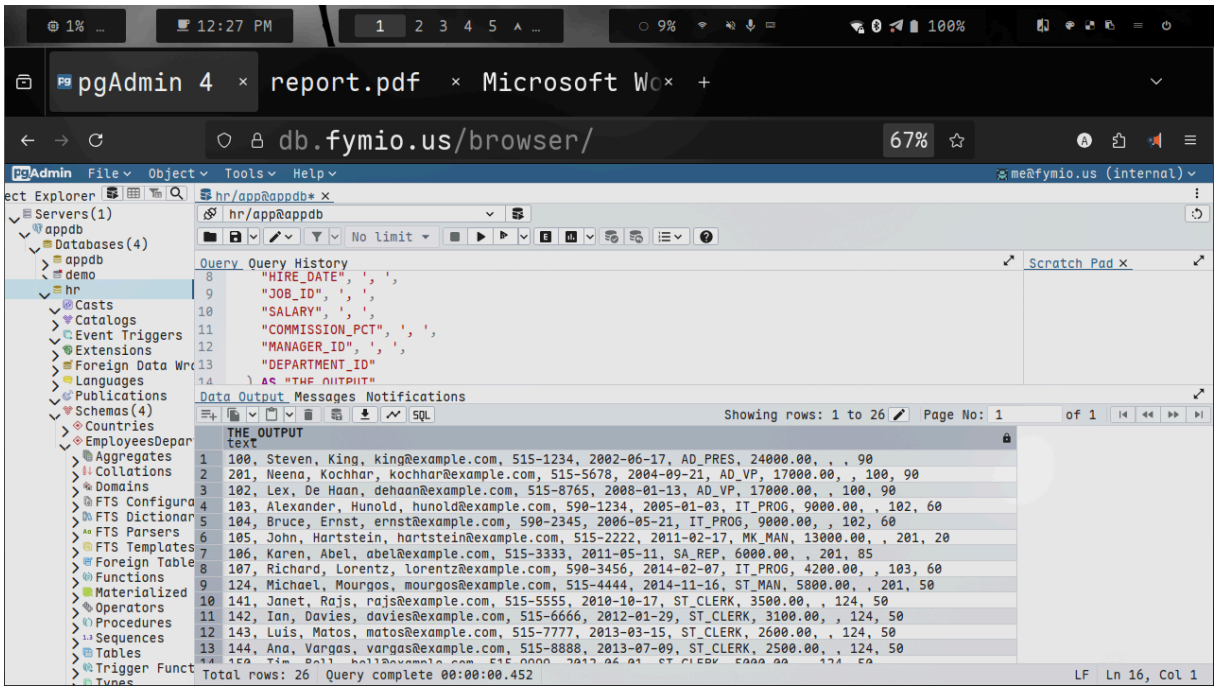

Рис. 22: Результат выполнения запроса.

4.2 Создайте запрос для вывода фамилий служащих (первая буква каждой фамилии должна быть заглавной, а остальные - строчными) и длину каждой фамилии для тех служащих, фамилия которых начинается с символа *J*, *A* или *M*. Присвойте соответствующие заголовки столбцам. Результат выполнения запроса должен соответствовать таблице 20.

Name	Length
“Abel”	4
“Almeida Castro”	14
“Alves Rocha”	11
“Matos”	5
“Mourgos”	7

Табл. 21: Результат выполнения запроса.

```sql
SELECT
  INITCAP("LAST_NAME") AS "Name",
  LENGTH("LAST_NAME") AS "Length"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE UPPER(SUBSTR("LAST_NAME", 1, 1)) IN ('J', 'A', 'M');
```

- `INITCAP("LAST_NAME")` делает первую букву фамилии заглавной, а остальные строчными;
- `LENGTH("LAST_NAME")` вычисляет количество символов в фамилии;
- `SUBSTR("LAST_NAME", 1, 1)` извлекает первый символ фамилии;
- `UPPER(...) IN ('J','A','M')` отбирает только тех сотрудников, чья фамилия начинается на *J*, *A* или *M* (без учёта регистра);
- Результат выводится в два столбца:
  - `Name` — фамилия с правильным регистром,
  - `Length` — длина фамилии.

Результат выполнения запроса представлен на рисунке 23.

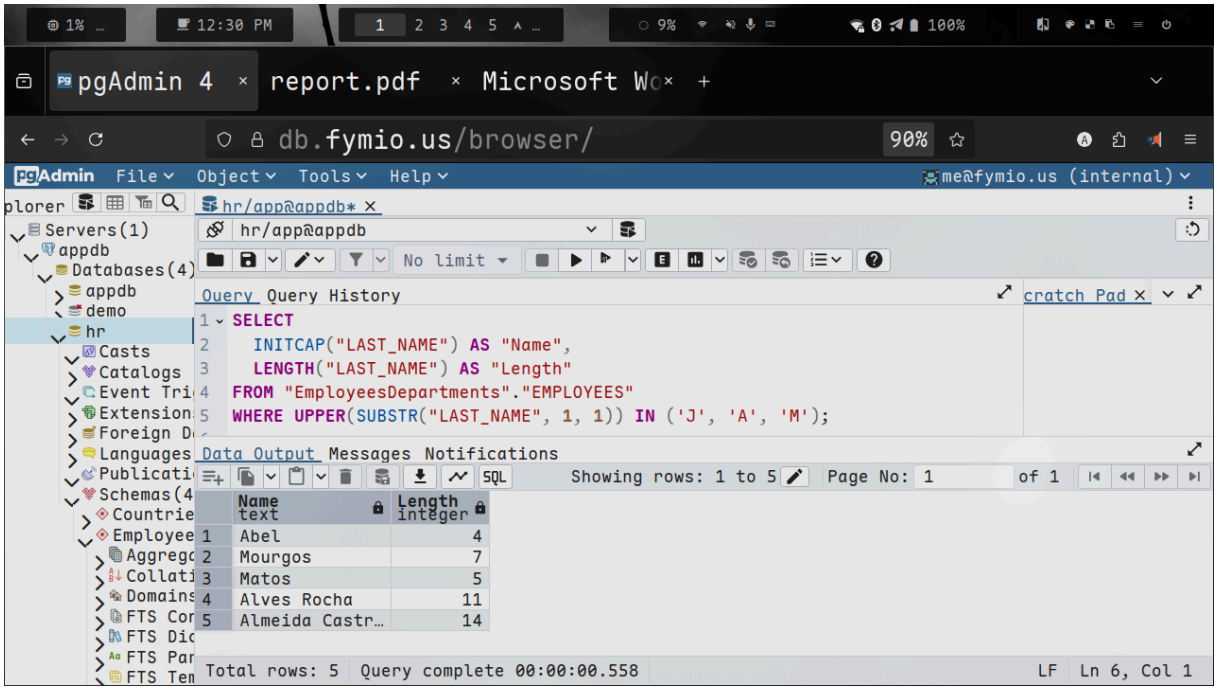

Рис. 23: Результат выполнения запроса.

4.3 Создайте запрос для вывода информации по каждому служащему в следующем виде: <фамилия> зарабатывает <оклад> в месяц, но желает <утроенный оклад>. Назовите столбец Dream Salaries. Результат запроса должен быть схож с таблицей 21.

Dream Salaries
King зарабатывает 24000 в месяц, но желает 72000
Kochhar зарабатывает 17000 в месяц, но желает 51000
De Haan зарабатывает 17000 в месяц, но желает 51000
Whalen зарабатывает 4400 в месяц, но желает 13200
Higgins зарабатывает 12000 в месяц, но желает 36000
…

Табл. 22: Результат выполнения запроса.

```
SELECT
  CONCAT(
    "LAST_NAME",
    ' зарабатывает ',
    "SALARY",
    ' в месяц, но желает ',
    "SALARY" * 3
  ) AS "Dream Salaries"
FROM "EmployeesDepartments"."EMPLOYEES";
```

- `CONCAT` объединяет несколько частей строки в одно выражение;
- `LAST_NAME` фамилия сотрудника;
- `SALARY` его текущий оклад;
- `"SALARY" * 3` утроенный оклад;
- все фрагменты объединяются в предложение.

Результат выполнения скрипта представлен на рисунке 24.

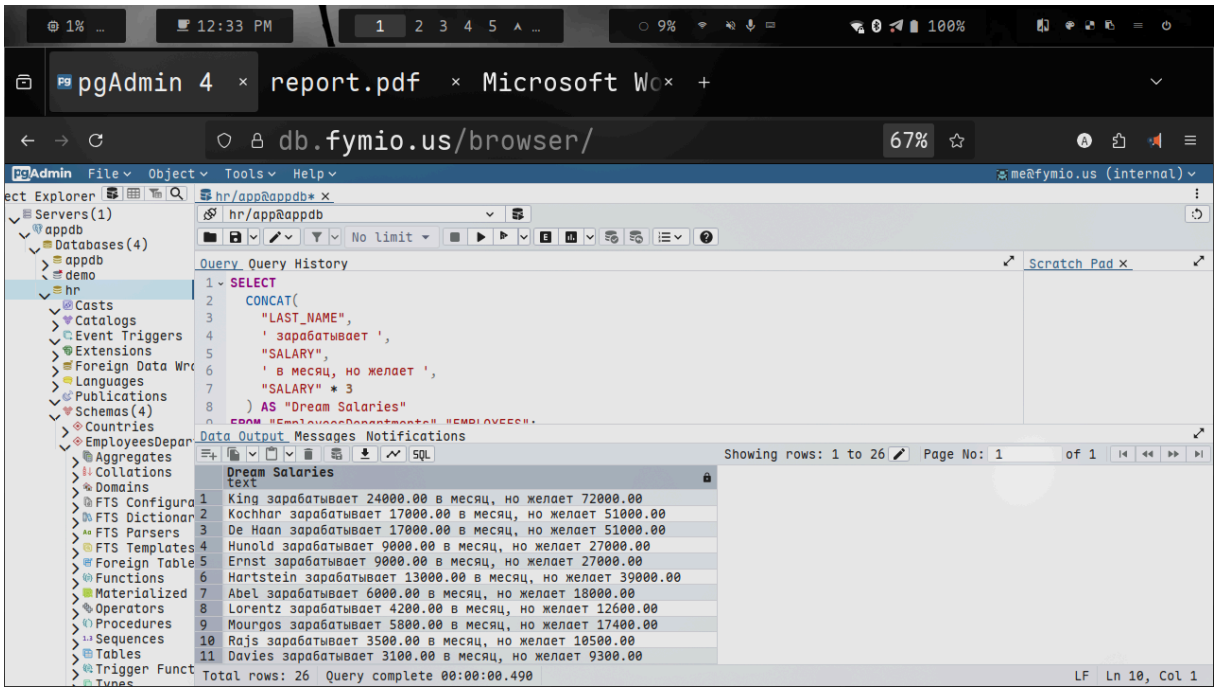

Рис. 24: Результат выполнения запроса.

Задание 5. Составление запросов, требующих функций для работы с датами и функции преобразования типов.

5.1 Напишите запрос для вывода текущей даты. Назовите столбец Date. Результат выполнения запроса должен быть подобен результату, представленному на таблице 22.

DATE
2024-10-20

Табл. 23: Результат выполнения запроса текущей даты.

```
SELECT CURRENT_DATE AS "Date";
```

- `CURRENT_DATE` стандартная функция, возвращающая текущую системную дату;
- `AS "Date"` задаёт имя столбца.

Результат выполнения запроса представлен на рисунке 25.

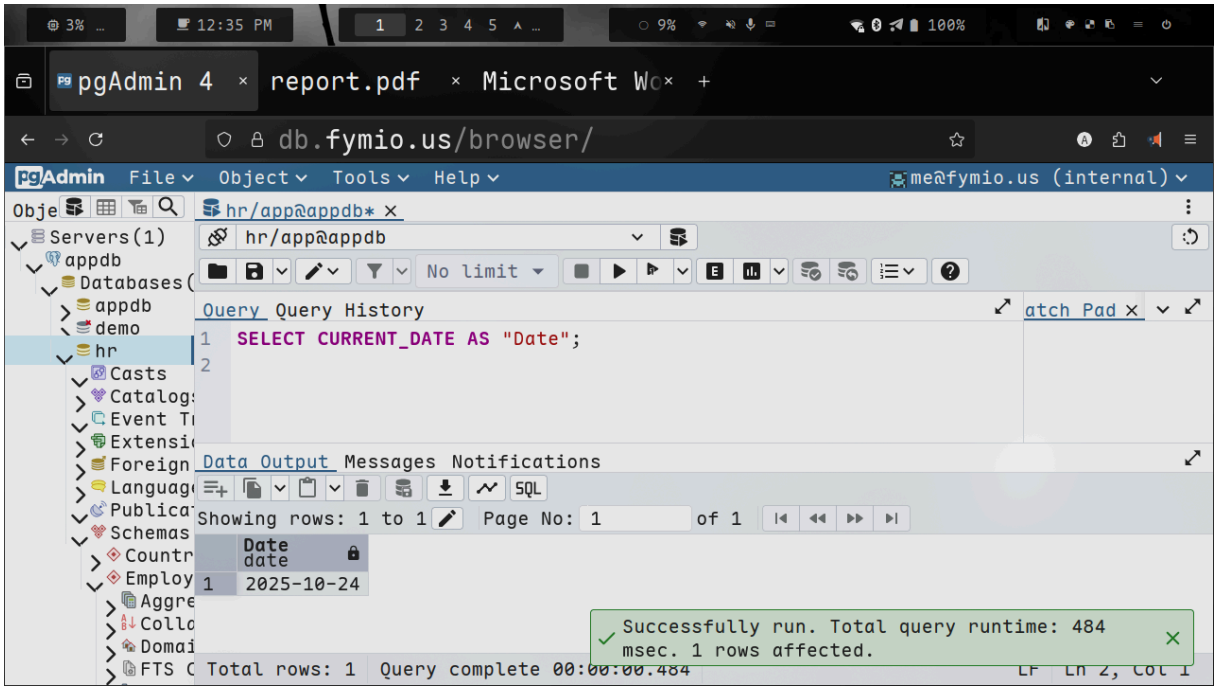

Рис. 25: Результат выполнения запроса.

5.2 Создайте запрос для вывода фамилии и даты найма всех служащих, нанятых в 2011 г. Результат выполнения запроса должен соответствовать таблице 23.

LAST_NAME	HIRE_DATE
Alves Rocha	2011-02-06
Hartstein	2011-02-17
Abel	2011-05-11
Hernandez	2011-06-13

Табл. 24: Результат выполнения запроса для служащих, нанятых в 2011г.

```
SELECT "LAST_NAME", "HIRE_DATE"
FROM "EmployeesDepartments"."EMPLOYEES"
WHERE EXTRACT(YEAR FROM "HIRE_DATE") = 2011;
```

- `EXTRACT(YEAR FROM "HIRE_DATE")` извлекает год из даты найма;
- `= 2011` выбирает только тех сотрудников, у которых год найма равен 2011.

Результат выполнения запроса представлен на рисунке 26.

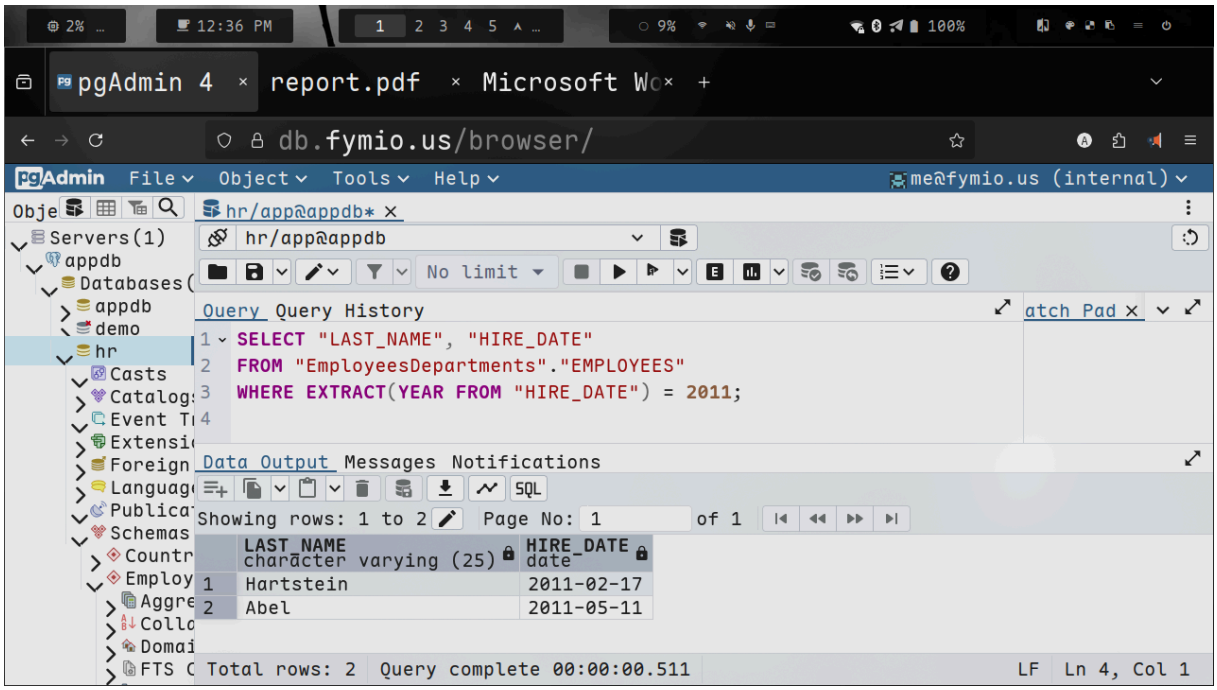

Рис. 26: Результат выполнения запроса.

5.3 Создайте запрос, который позволяет для каждого служащего вывести фамилию, дату найма и вычислят количество месяцев со дня найма до настоящего времени. Назовите столбец MONTH_WORKED. Результаты отсортируйте по количеству отработанных месяцев. Результат выполнения запроса должен быть схож с таблицей 24.

LAST_NAME	HIRE_DATE	MONTH_WORKED
Safwah	1997-01-06	345
King	2002-06-17	280
Whalen	2002-09-17	277
Kochhar	2004-09-21	252
Steiner	2004-11-02	251
Hunold	2005-01-03	249
Ernst	2006-05-21	232
Reinhard	2007-07-25	218
...	...	...

Табл. 25: Результат выполнения запроса.

```
SELECT
  "LAST_NAME",
  "HIRE_DATE",
  (EXTRACT(YEAR FROM AGE(CURRENT_DATE, "HIRE_DATE")) * 12
   + EXTRACT(MONTH FROM AGE(CURRENT_DATE, "HIRE_DATE"))) AS "MONTH_WORKED"
FROM "EmployeesDepartments"."EMPLOYEES"
ORDER BY "MONTH_WORKED" DESC;
```

- `AGE(CURRENT_DATE, "HIRE_DATE")` — возвращает интервал между двумя датами;

- `EXTRACT(YEAR FROM AGE(...)) * 12 + EXTRACT(MONTH FROM AGE(...))` переводит годы и месяцы в общее количество месяцев;
- `ORDER BY "MONTH_WORKED" DESC` сортирует по количеству месяцев в порядке убывания.

Результат выполнения запроса представлен на рисунке 27.

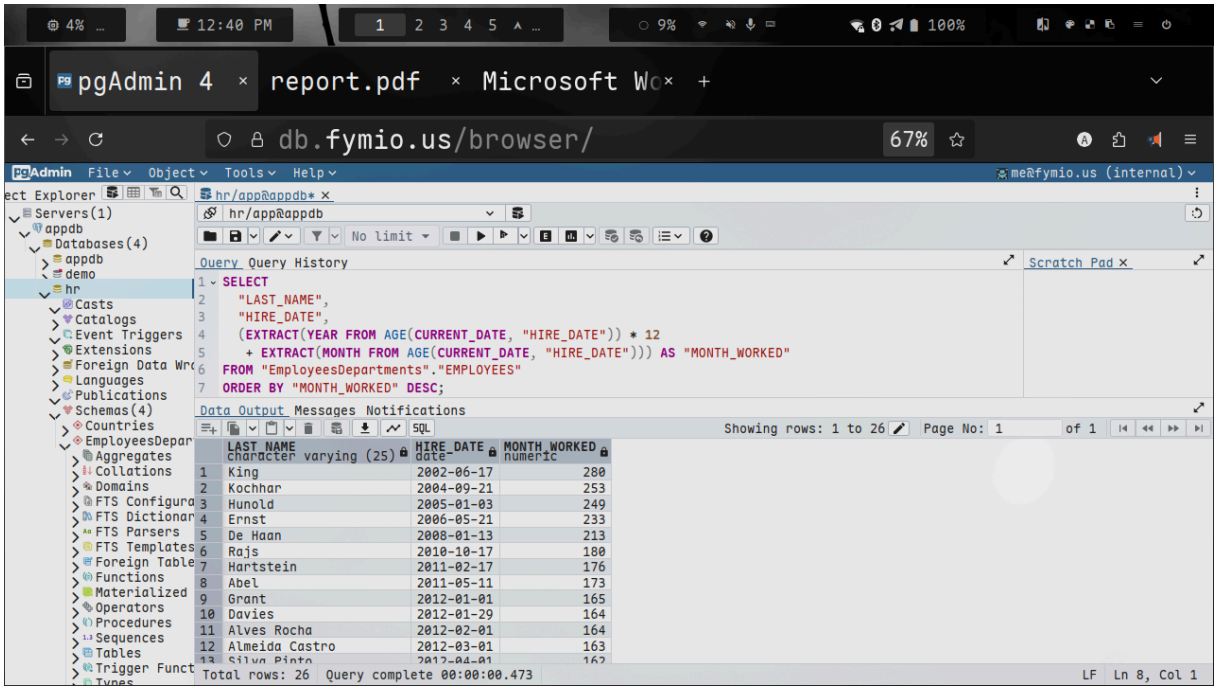

Рис. 27: Результат выполнения запроса.

5.4 Создайте запрос, который позволяет для каждого служащего вывести фамилию, дату найма и день недели, когда он был нанят на работу. Назовите последний столбец DAY. Отсортируйте результаты по датам. Результат выполнения запроса должен быть схож с таблицей 25.

LAST_NAME	HIRE_DATE	DAY
Stocks	2015-12-16	WEDNESDAY
Newton	2015-12-16	WEDNESDAY
Heiden	2015-07-06	MONDAY
Ricci	2015-05-17	SUNDAY
Zlotkey	2015-01-29	THURSDAY
Mourgos	2014-11-16	SUNDAY
Grant	2014-05-24	SATURDAY
Bell	2014-04-01	TUESDAY
Lorentz	2014-02-07	FRIDAY
...	...	...

Табл. 26: Результат выполнения запроса.

```
SELECT
  "LAST_NAME",
  "HIRE_DATE",
  TO_CHAR("HIRE_DATE", 'FMDay') AS "DAY"
```

```
FROM "EmployeesDepartments"."EMPLOYEES"
ORDER BY "HIRE_DATE";
```

- `TO_CHAR("HIRE_DATE", 'FMDay')` преобразует дату в название дня недели;
  - `FM` убирает лишние пробелы в начале и конце;
  - возвращается день недели полностью;
- столбец переименован в `DAY`;
- `ORDER BY "HIRE_DATE"` сортирует результаты по дате найма, от ранних к поздним.

Результат выполнения запроса представлен на рисунке 28.

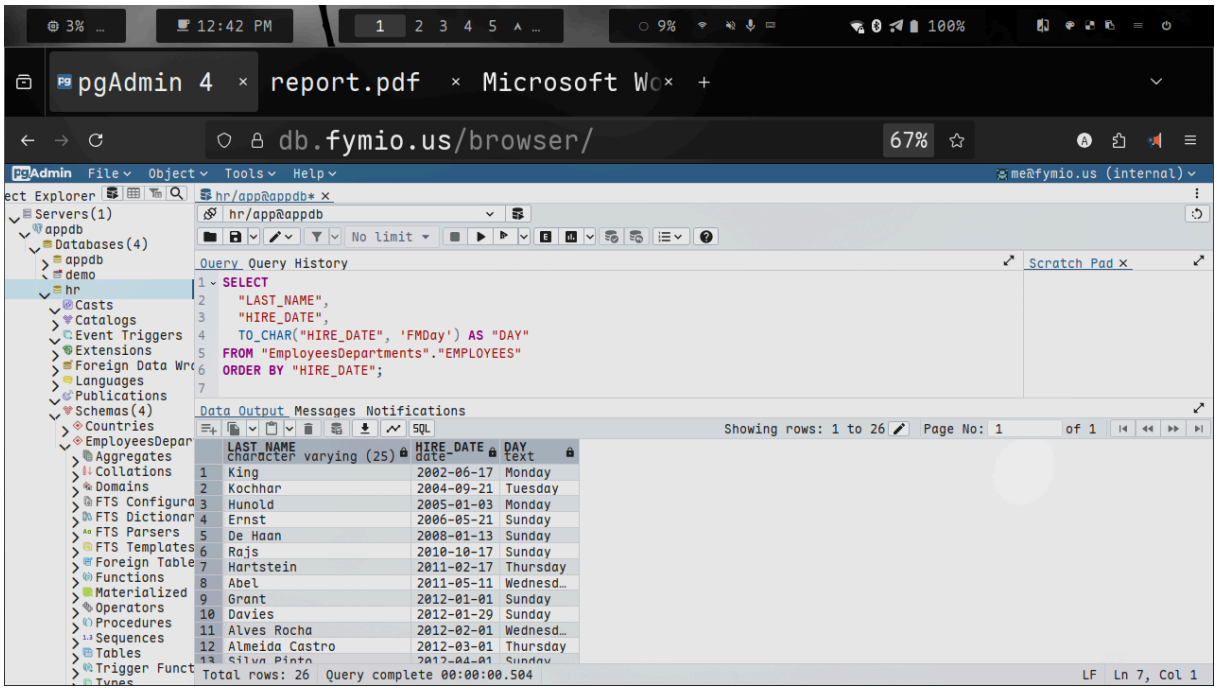

Рис. 28: Результат выполнения запроса.

Задание 6. Составление запросов, требующих применения условных выражений.

6.1 Создайте запрос, который позволяет для каждого служащего вывести фамилию и сумму комиссионных. Если служащий не зарабатывает комиссионных, укажите в столбце “No Commission”. Назовите столбец COMM. Используются ф-ции преобразования типов и условное выражение COALESCE. Результат выполнения запроса должен быть схож с таблицей 25.

LAST_NAME	COMM
King	No Commission
Kochhar	No Commission
De Haan	No Commission
Hunold	No Commission
Ernst	No Commission
Lorentz	No Commission
Mourgos	No Commission
Rajs	No Commission
Davies	No Commission
Matos	No Commission
Vargas	No Commission
Zlotkey	0.20
Abel	0.30
…	…

Табл. 27: Результат выполнения запроса.

```
SELECT
  "LAST_NAME",
  COALESCE(TO_CHAR("COMMISSION_PCT", 'FM0.00'), 'No Commission') AS "COMM"
FROM "EmployeesDepartments"."EMPLOYEES";
```

- `TO_CHAR("COMMISSION_PCT", 'FM0.00')` преобразует числовое значение комиссионных в текстовый формат с двумя десятичными знаками;
- `COALESCE(..., 'No Commission')` если значение комиссионных равно NULL, выводится строка ‘No Commission’;
- столбец переименован в `COMM`;
- вывод показывает фамилию сотрудника и либо его комиссионные, либо надпись ‘No Commission’.

Результат выполнения запроса представлен на рисунке 29.

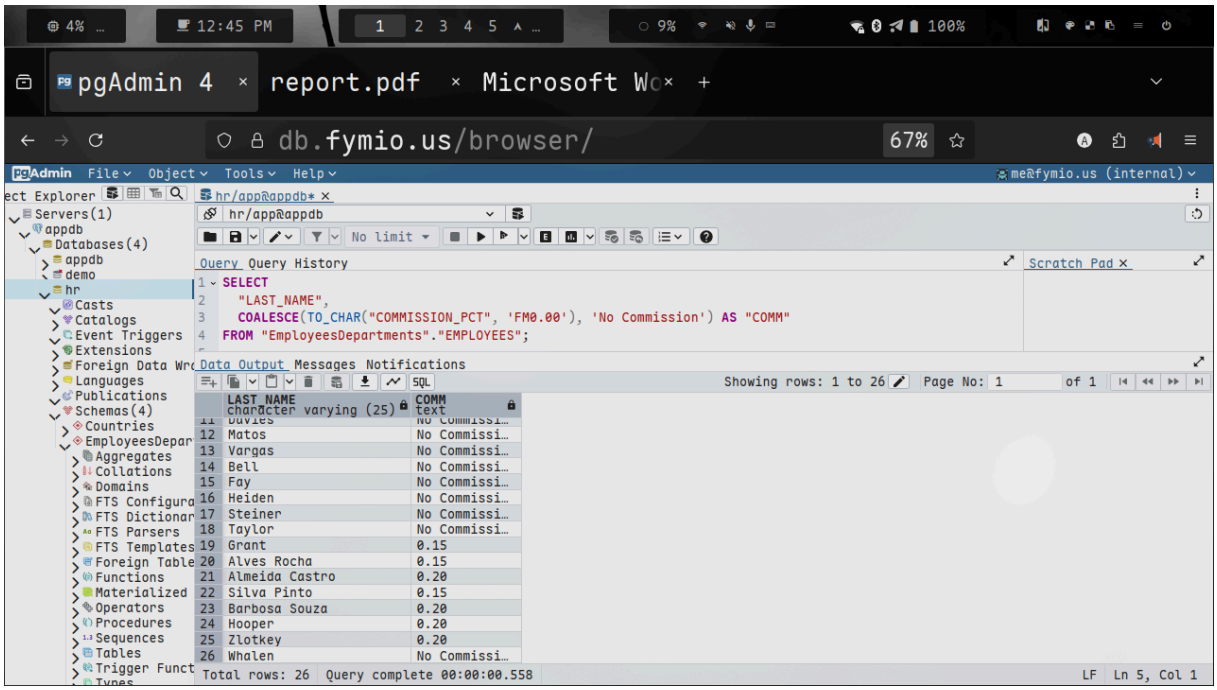

Рис. 29: Результат выполнения запроса.

6.2 Создайте запрос, который позволяет для каждого служащего вывести должность и её разряда (grade), используя условное выражение CASE. Разряд каждого типа должности JOB_ID приведён в таблице 26, а результат выполнения запроса должен быть схож с таблицей 27.

Должность	Разряд
AD_PRES	E
ST_MAN	D
IT_PROG	C
SA_REP	B
ST_CLERK	A
Другая	0

Табл. 28: Соответствие каждого типа должности и разряда.

LAST_NAME	JOB_ID	GRADE
King	AD_PRES	E
Kochhar	AD_VP	0
De Haan	AD_VP	0
Whalen	AD_ASST	0
Higgins	AC_MGR	0
Gietz	AC_ACCOUNT	0
Zlotkey	SA_MAN	0
Abel	SA_REP	B
Taylor	SA_REP	B
Grant	SA_REP	B
Mourgos	ST_MAN	D
Rajs	ST_CLERK	A
King	AD_PRES	E
Kochhar	AD_VP	0
De Haan	AD_VP	0
Whalen	AD_ASST	0
…	…	…

Табл. 29: Результат выполнения запроса.

```
SELECT
  "JOB_ID" AS "Должность",
  CASE "JOB_ID"
    WHEN 'AD_PRES' THEN 'E'
    WHEN 'ST_MAN'  THEN 'D'
    WHEN 'IT_PROG' THEN 'C'
    WHEN 'SA_REP'  THEN 'B'
    WHEN 'ST_CLERK' THEN 'A'
    ELSE '0'
  END AS "GRADE"
FROM "EmployeesDepartments"."EMPLOYEES";
```

- `CASE ... WHEN ... THEN ... ELSE ... END` позволяет для каждого типа должности присвоить соответствующий разряд;
- должности, не указанные в таблице 26, получают разряд ‘0’ через `ELSE '0'`;
- столбцы переименованы в `Должность` и `GRADE`;

Результат выполнения запроса представлен на рисунке 30.

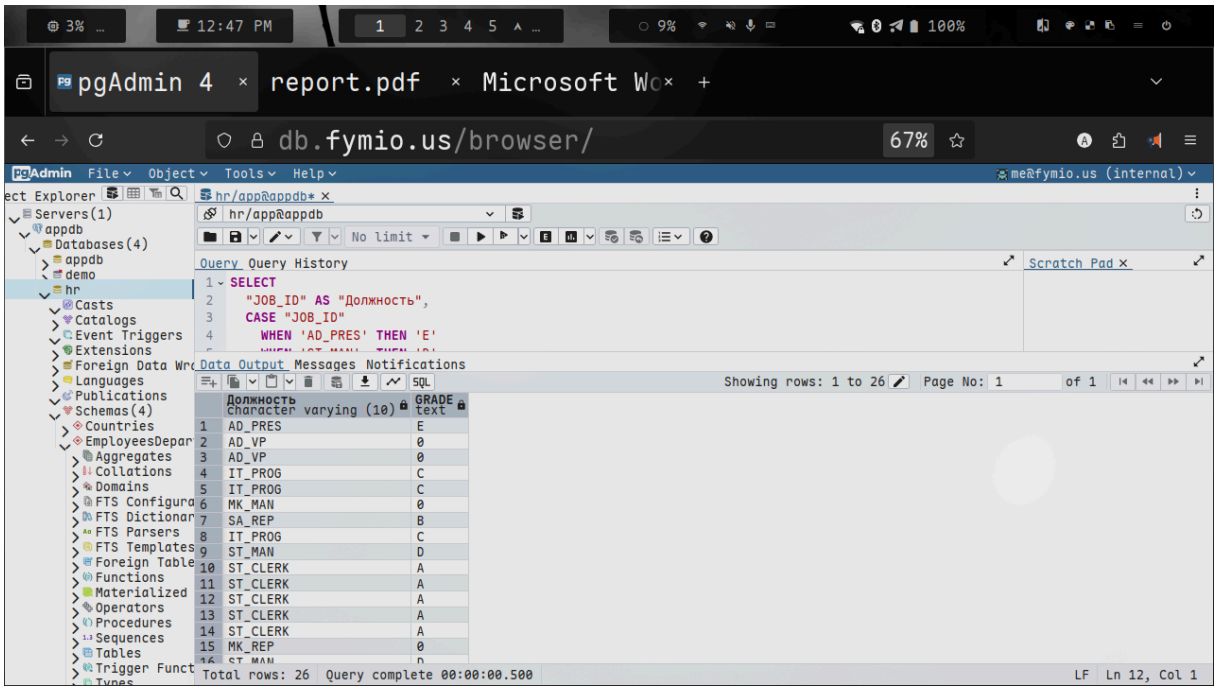

Рис. 30: Результат выполнения запроса.

Выводы и анализ результатов работы

В ходе выполнения работы были успешно освоены основные конструкции языка SQL для работы с реляционными базами данных. Все поставленные задачи выполнены, запросы составлены корректно и возвращают ожидаемые результаты.

- Научился формировать запросы на выборку данных с использованием оператора `SELECT`
- Освоил использование предложений `WHERE` для фильтрации данных по различным условиям
- Изучил применение `ORDER BY` для сортировки результатов запросов
- Научился использовать `DISTINCT` для выборки уникальных значений
- Освоил создание псевдонимов для столбцов с помощью `AS`
- Применил операторы сравнения
- Использовал логические операторы
- Освоил работу с операторами `BETWEEN`, `IN`, `NOT IN` для проверки диапазонов и списков значений
- Научился работать с `NULL` значениями через `IS NULL` и `IS NOT NULL`
- Применил оператор `LIKE` для поиска по шаблону в текстовых данных
- Освоил функции округления `ROUND()` для математических вычислений
- Научился выполнять арифметические операции в запросах (сложение, вычитание, умножение, деление)
- Создавал вычисляемые поля для расчёта новых значений на основе существующих данных
- Применил функцию `CONCAT()` для объединения нескольких строковых значений
- Использовал `INITCAP()` для форматирования текста с заглавной первой буквой
- Освоил `LENGTH()` для определения длины строк
- Применил `SUBSTR()` для извлечения подстрок
- Научился использовать оператор конкатенации `||` в PostgreSQL
- Освоил функцию `CURRENT_DATE` для получения текущей системной даты
- Применил `EXTRACT()` для извлечения компонентов даты

- Использовал функцию `AGE()` для вычисления разницы между датами
- Научился форматировать даты с помощью `TO_CHAR()` для вывода дней недели
- Выполнил сложные вычисления с датами, такие как расчёт количества отработанных месяцев
- Освоил функцию `COALESCE()` для обработки `NULL` значений и замены их на заданные значения
- Применил конструкцию `CASE ... WHEN ... THEN ... ELSE ... END` для создания условной логики непосредственно в запросах
- Научился создавать сложные вычисляемые поля с использованием множественных условий
- Развил навыки поиска и исправления синтаксических ошибок в SQL-запросах
- Научился анализировать структуру таблиц через `information_schema.columns`
- Понял важность правильного использования кавычек для регистрозависимых имён в PostgreSQL
- Освоил методику пошаговой разработки сложных запросов

В процессе выполнения работы возникли следующие сложности:

- В PostgreSQL при создании объектов с кавычками имена становятся регистрозависимыми. Решение: последовательное использование двойных кавычек для всех идентификаторов таблиц и столбцов.
- Особенности сравнения и обработки NULL потребовали использования специальных конструкций. Решение: изучение документации и применение соответствующих функций.
- Получение названий дней недели на английском языке потребовало использования функции `TO_CHAR()` с правильным форматом. Решение: применение модификатора `FM` для удаления лишних пробелов.
- Расчёт количества месяцев между датами потребовал комбинирования функций `AGE()` и `EXTRACT()`. Решение: декомпозиция задачи на вычисление отдельно лет и месяцев с последующим их объединением.

В результате выполнения практической работы была достигнута поставленная цель: освоены основные конструкции SQL и получены практические навыки составления запросов различной сложности для работы с реляционными базами данных.