

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

Факультет инфокоммуникационных технологий

Направление подготовки 11.03.02

Практическая работа №3

Выполнил:

Дощенников Никита Андреевич

Группа: К3221

Проверила:

Татьяна Евгеньевна Войтюк

Санкт-Петербург
2025

Цель работы

Целью данной работы является изучение основных операций с базами данных на примере postgresql. В ходе работы мы учимся выполнять различные виды соединений таблиц, использовать агрегатные функции, группировать данные и получать сводную информацию о сотрудниках и отделах. Всё это позволяет лучше понять, как организуются данные в реальных информационных системах и как из них можно получать нужную информацию.

Задачи, решаемые при выполнении работы

В рамках работы необходимо выполнить несколько типов заданий. Сначала мы работаем с объединением таблиц, чтобы научиться получать информацию о сотрудниках, их отделах, руководителях и коллегах. Затем изучаем групповые функции, которые позволяют подсчитать количество сотрудников, средние и максимальные зарплаты, разницу окладов и другие показатели. Также нужно уметь фильтровать данные по различным условиям, сортировать результаты и использовать агрегирование, чтобы формировать отчёты по отделам и должностям.

Исходные данные

Для выполнения работы использовалась база данных, содержащая таблицы с информацией о сотрудниках, отделах, местоположениях, странах и регионах. Таблицы включают такие поля, как имя, фамилия, идентификатор сотрудника, должность, зарплата, бонус, дата найма, идентификатор отдела и руководителя. Эти данные позволяют выполнять все необходимые запросы: соединять таблицы, группировать и фильтровать информацию, а также анализировать структуру и показатели работы сотрудников и отделов.

Выполнение работы

Задание 1. Использование объединения таблиц

1.1

Напишите запрос для вывода фамилии, имени, названия отдела для всех работников, в фамилии которых есть буква «и» (в строчном регистре). Укажите, какой тип соединения таблиц

используется в данном запросе, и задокументируйте результат его выполнения.

```
SELECT
    e."LAST_NAME",
    e."FIRST_NAME",
    d."DEPARTMENT_NAME"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
WHERE e."LAST_NAME" LIKE '%u%';
```

- Соединяем таблицы `EMPLOYEES` и `DEPARTMENTS` по `DEPARTMENT_ID` с помощью `INNER JOIN`.
- Выбираем фамилию, имя и название отдела.
- Фильтруем только тех сотрудников, у которых фамилия содержит букву «и».

```
Object Explorer      appdb/appWeb PostgreSQL*
Tables(6)
  > DEPARTMENTS
  > EMPLOYEES
  > JOBS
  .> JOB GRADES
    > Columns(3)
      > GRADE_LEVEL
      > LOWEST_SAL
      > HIGHEST_SAL
    > Constraints
    > Indexes
    > RLS Policies
    > Rules
    > Triggers
  > JOB_HISTORY
  > LOCATIONS
  > Trigger Functions
  > Types
  > Views
  > countries

Query   Query History
4  d."DEPARTMENT_NAME"
5  FROM "EmployeesDepartments"."EMPLOYEES" e
6  JOIN "EmployeesDepartments"."DEPARTMENTS" d
7  ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
8  WHERE e."LAST_NAME" LIKE '%u%';

Data Output  Messages  Notifications
Showing rows: 1 to 3  Page No: 1 of 1
LAST_NAME  FIRST_NAME  DEPARTMENT_NAME
character varying (25)  character varying (20)  character varying (25)
1 Hunold  Alexander  IT
2 Mourgos
3 Barbosa Souza

Total rows: 3  Query complete 00:00:00.261  LF  Ln 8, Col 32
```

Рис. 1: Результат выполнения запроса задания 1.1

1.2

Напишите запрос для вывода имени, фамилии, названия должности и названия отдела для всех работников. Отсортируйте результат по идентификатору работника. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```

SELECT
    e."FIRST_NAME",
    e."LAST_NAME",
    e."JOB_ID",
    d."DEPARTMENT_NAME"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
ORDER BY e."EMPLOYEE_ID";

```

- Соединяем EMPLOYEES и DEPARTMENTS по DEPARTMENT_ID через INNER JOIN.
- Выбираем имя, фамилию, должность и отдел сотрудника.
- Сортируем по EMPLOYEE_ID для просмотра всех сотрудников по порядку.

```

SELECT
    e."FIRST_NAME",
    e."LAST_NAME",
    e."JOB_ID",
    d."DEPARTMENT_NAME"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
ORDER BY e."EMPLOYEE_ID";

```

	FIRST_NAME	LAST_NAME	JOB_ID
1	Steven	King	AD_PRES
2	Lex		
3	Alexander		
			Successfully run. Total query runtime: 186 msec. 26 rows affected.
	Total rows: 26	Query complete 00:00:00.186	LF Ln 9, Col 26

Рис. 2: Результат выполнения запроса задания 1.2

1.3

Напишите запрос для вывода названия отдела, фамилии и имени сотрудника для всех сотрудников, у которых есть бонус, работающих в 80-ом и 85-ом отделах. Полученный результат отсортируйте по номеру отдела, размеру бонуса по убыванию, а затем фамилии. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```

SELECT
    d."DEPARTMENT_NAME",
    e."LAST_NAME",
    e."FIRST_NAME",
    e."COMMISSION_PCT" AS "Bonus"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
WHERE e."COMMISSION_PCT" IS NOT NULL
    AND e."DEPARTMENT_ID" IN (80, 85)
ORDER BY e."DEPARTMENT_ID", e."COMMISSION_PCT" DESC,
    e."LAST_NAME";

```

- Выбираем сотрудников из отделов 80 и 85 с ненулевым бонусом.
- Соединяем таблицы сотрудников и отделов через `INNER JOIN`.
- Сортируем по номеру отдела, бонус по убыванию, фамилия.
- В результате видны только сотрудники с бонусом и существующим отделом.

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer pane displays the database schema with tables like DEPARTMENTS, EMPLOYEES, and JOBS. The central pane contains the SQL query editor with the following code:

```

SELECT
    d."DEPARTMENT_NAME",
    e."LAST_NAME",
    e."FIRST_NAME",
    e."COMMISSION_PCT" AS "Bonus"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
WHERE e."COMMISSION_PCT" IS NOT NULL
    AND e."DEPARTMENT_ID" IN (80, 85)
ORDER BY e."DEPARTMENT_ID", e."COMMISSION_PCT" DESC,
    e."LAST_NAME";

```

The right pane shows the Data Output tab displaying the results of the query:

	DEPARTMENT_NAME	LAST_NAME	FIRST_NAME
1	Sales - Europe	Almeida Castro	Carlos
2	Sales - Europe		
3	Sales - Europe		

Below the table, a message indicates the query was successfully run and completed in 251 msec, with 7 rows affected. The total number of rows is 7.

Рис. 3: Результат выполнения запроса задания 1.3

1.4

Напишите запрос для вывода фамилии, имени, названия страны и региона для всех работников, работающих в Северной Америке. Отсортируйте результат по названию страны и фамилии сотрудника. Укажите, какой тип соединения таблиц

используется в данном запросе, и задокументируйте результат его выполнения.

```
SELECT
    e."LAST_NAME",
    e."FIRST_NAME",
    c."COUNTRY_NAME",
    r."REGION_NAME"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
JOIN "EmployeesDepartments"."LOCATIONS" l
    ON d."LOCATION_ID" = l."LOCATION_ID"
JOIN "Countries"."COUNTRIES" c
    ON l."COUNTRY_ID" = c."COUNTRY_ID"
JOIN "Countries"."REGIONS" r
    ON c."REGION_ID" = r."REGION_ID"
WHERE r."REGION_NAME" = 'Americas'
ORDER BY c."COUNTRY_NAME", e."LAST_NAME";
```

- Соединяем сотрудников, отделы, локации, страны, регионы
- Используем INNER JOIN
- Фильтруем регион Americas
- Сортируем по стране и фамилии сотрудника

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with tables like DEPARTMENTS, EMPLOYEES, JOBS, and COLUMNS.
- Query Pad:** Contains the SQL query provided in the text block.
- Data Output:** Displays the results of the query execution:
 - LAST_NAME character vary
 - Successfully run. Total query runtime: 175 msec. 0 rows affected.
 - Total rows: 0 Query complete 00:00:00.175

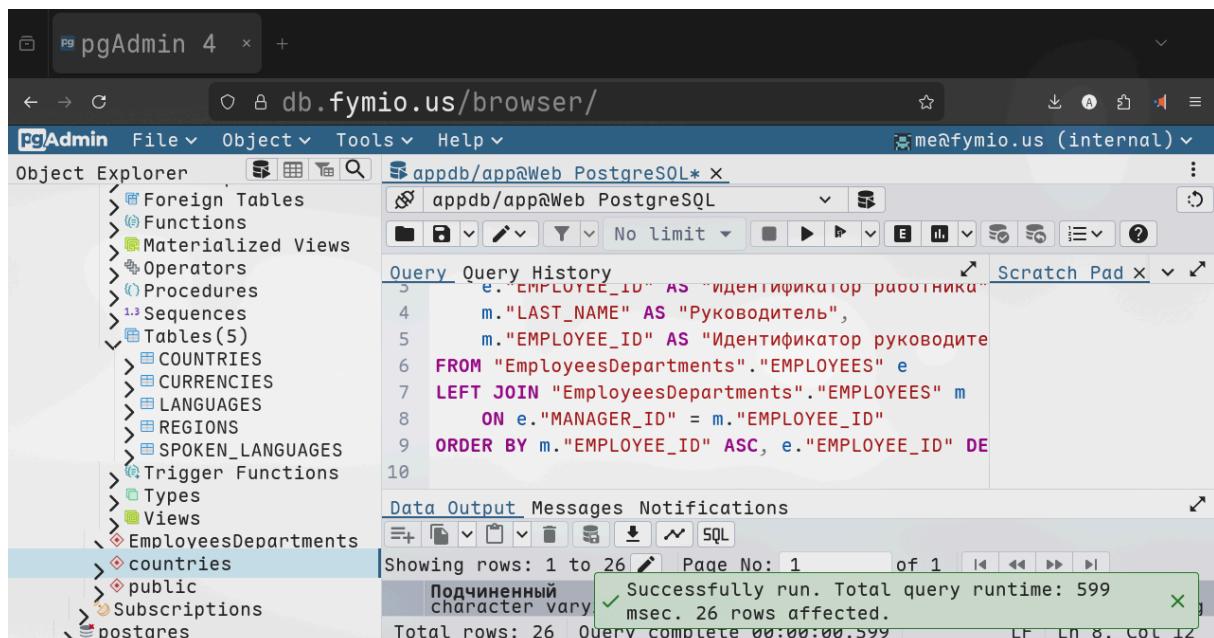
Рис. 4: Результат выполнения запроса задания 1.4

1.5

Напишите запрос для вывода фамилии и идентификатора работника, а также фамилии и идентификатора его начальника. Назовите столбцы результата «Подчиненный», «Идентификатор работника», «Руководитель», «Идентификатор руководителя». Отсортируйте результат по идентификатору руководителя по возрастанию и по идентификатору работника по убыванию. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```
SELECT
    e."LAST_NAME" AS "Подчиненный",
    e."EMPLOYEE_ID" AS "Идентификатор работника",
    m."LAST_NAME" AS "Руководитель",
    m."EMPLOYEE_ID" AS "Идентификатор руководителя"
FROM "EmployeesDepartments"."EMPLOYEES" e
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" m
    ON e."MANAGER_ID" = m."EMPLOYEE_ID"
ORDER BY m."EMPLOYEE_ID" ASC, e."EMPLOYEE_ID" DESC;
```

- Используется `LEFT JOIN`, чтобы включить всех сотрудников, даже если у них нет начальника.
- Сортировка сначала по ID менеджера, потом по ID сотрудника.



```
SELECT
    e."LAST_NAME" AS "Подчиненный",
    e."EMPLOYEE_ID" AS "Идентификатор работника",
    m."LAST_NAME" AS "Руководитель",
    m."EMPLOYEE_ID" AS "Идентификатор руководителя"
FROM "EmployeesDepartments"."EMPLOYEES" e
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" m
    ON e."MANAGER_ID" = m."EMPLOYEE_ID"
ORDER BY m."EMPLOYEE_ID" ASC, e."EMPLOYEE_ID" DESC;
```

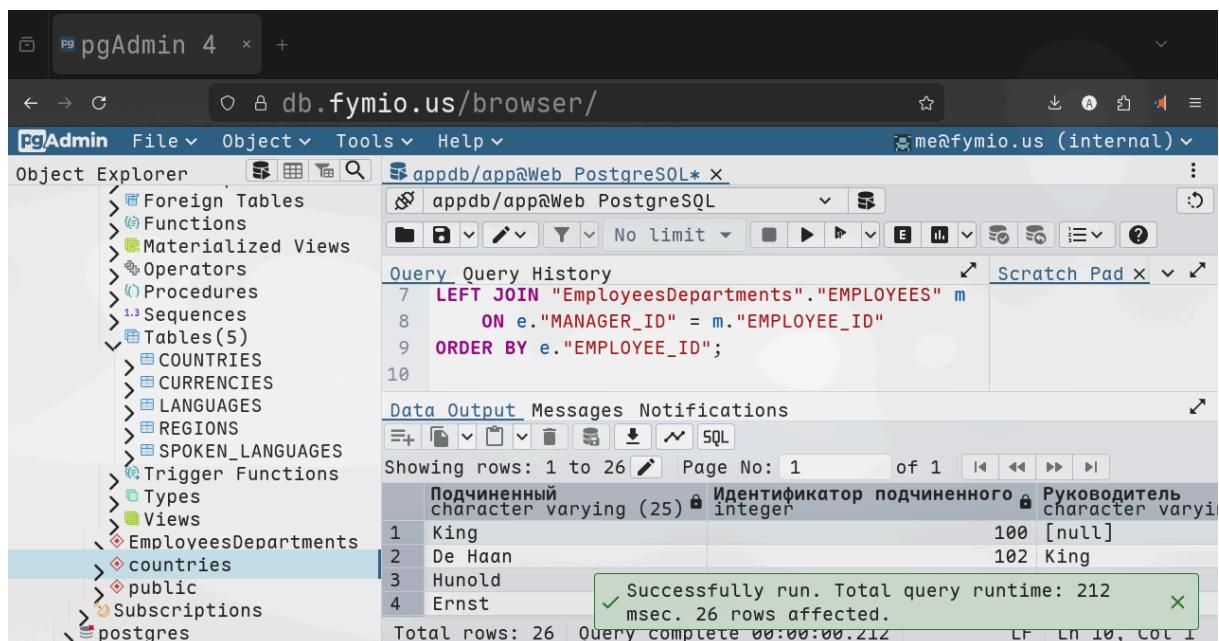
Рис. 5: Результат выполнения запроса задания 1.5

1.6

Измените запрос из пункта 1.5. таким образом, чтобы получить фамилии всех работников в столбце «Подчиненный», включая Кинга, который не имеет руководителя. Отсортируйте результат по идентификатору подчиненного, укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```
SELECT
    e."LAST_NAME" AS "Подчиненный",
    e."EMPLOYEE_ID" AS "Идентификатор подчиненного",
    m."LAST_NAME" AS "Руководитель",
    m."EMPLOYEE_ID" AS "Идентификатор руководителя"
FROM "EmployeesDepartments"."EMPLOYEES" e
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" m
    ON e."MANAGER_ID" = m."EMPLOYEE_ID"
ORDER BY e."EMPLOYEE_ID";
```

- Используем `LEFT JOIN` таблицы `EMPLOYEES`, чтобы получить и менеджеров, и всех подчинённых, включая тех, у кого нет руководителя.
- Сортировка выполняется по ID подчинённого для удобного просмотра.



Подчиненный	Идентификатор подчиненного	Руководитель
King	100	[null]
De Haan	102	King
Hunold		
Ernst		

Рис. 6: Результат выполнения запроса задания 1.6

1.7

Напишите запрос для вывода названия отдела, фамилии сотрудника и фамилий всех его коллег для сотрудников Fay, Hartstein и Davies. Назовите столбцы результата «Отдел», «Работник», «Коллеги». Отсортируйте результат по отделу и фамилии сотрудника. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```
SELECT
    d."DEPARTMENT_NAME" AS "Отдел",
    e1."LAST_NAME" AS "Работник",
    STRING_AGG(e2."LAST_NAME", ', ') AS "Коллеги"
FROM "EmployeesDepartments"."EMPLOYEES" e1
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e1."DEPARTMENT_ID" = d."DEPARTMENT_ID"
JOIN "EmployeesDepartments"."EMPLOYEES" e2
    ON e1."DEPARTMENT_ID" = e2."DEPARTMENT_ID"
        AND e1."EMPLOYEE_ID" <> e2."EMPLOYEE_ID"
WHERE e1."LAST_NAME" IN ('Fay', 'Hartstein', 'Davies')
GROUP BY d."DEPARTMENT_NAME", e1."LAST_NAME"
ORDER BY d."DEPARTMENT_NAME", e1."LAST_NAME";
```

- Используем `JOIN`, чтобы найти коллег по одному отделу, исключая самого сотрудника.
- Применяем `STRING_AGG` для объединения всех фамилий коллег в одну строку.
- Фильтруем только выбранных сотрудников и сортируем результат по отделу и фамилии.

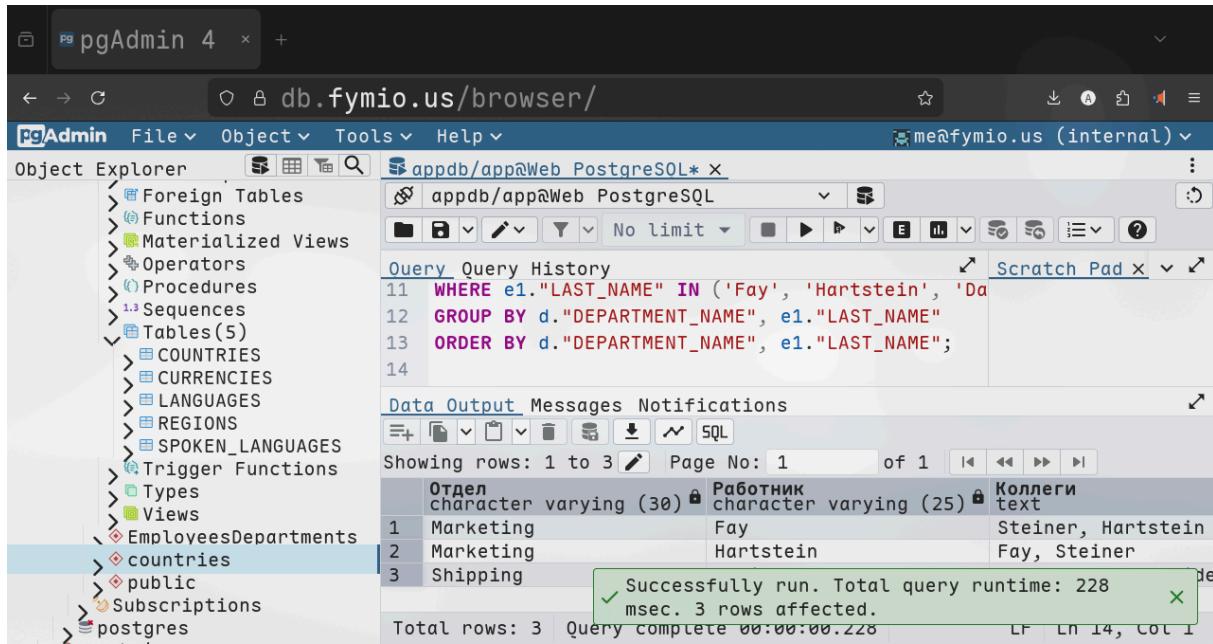


Рис. 7: Результат выполнения запроса задания 1.7

1.8

Напишите запрос для вывода всех категорий работников (GRADE_LEVEL), их фамилий, размеров оклада, названий должностей и названий отделов. Если в некоторой категории нет работников, то эта категория всё равно должна присутствовать в результате. Отсортируйте результат по категории работника, отделу и фамилии. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```

SELECT
    jg."GRADE_LEVEL" AS "Категория",
    e."LAST_NAME" AS "Фамилия",
    e."SALARY" AS "Оклад",
    e."JOB_ID" AS "Должность",
    d."DEPARTMENT_NAME" AS "Отдел"
FROM "EmployeesDepartments"."JOB_GRADES" jg
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON e."SALARY" BETWEEN jg."LOWEST_SAL" AND jg."HIGHEST_SAL"
LEFT JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
ORDER BY jg."GRADE_LEVEL", d."DEPARTMENT_NAME", e."LAST_NAME";

```

- Основная таблица — `JOB_GRADES`.

- Сотрудники и отделы присоединяются через `LEFT JOIN`.
- Таким образом, категории без работников всё равно будут отображаться.
- Результат отсортирован по категории, отдел, фамилия.

```

pgAdmin 4 + db.fymio.us/browser/
pgAdmin File Object Tools Help
Object Explorer Sequences Tables(6)
  DEPARTMENTS
  EMPLOYEES
  JOBS
  JOB GRADES
    Columns(3)
      GRADE_LEVEL
      LOWEST_SAL
      HIGHEST_SAL
    Constraints
    Indexes
    RLS Policies
    Rules
    Triggers
  JOB_HISTORY
  LOCATIONS
  Trigger Functions
  Types
  Views
  countries
appdb/app@Web PostgreSQL*
Query History
SELECT e."JOB_ID" AS "должность",
       d."DEPARTMENT_NAME" AS "Отдел"
FROM "EmployeesDepartments"."JOB_GRADES" jg
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
ON e."SALARY" BETWEEN jg."LOWEST_SAL" AND jg
LEFT JOIN "EmployeesDepartments"."DEPARTMENTS" d
ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
ORDER BY jg."GRADE_LEVEL", d."DEPARTMENT_NAME",
13
Data Output Messages Notifications
Категория character vary
Successfully run. Total query runtime: 216 msec. 0 rows affected.
Total rows: 0 Query complete 00:00:00.216 LF Ln 10, Col 32

```

Рис. 8: Результат выполнения запроса задания 1.8

1.9

Напишите запрос для вывода фамилий и дат найма всех сотрудников, а также фамилий и дат найма их руководителей, для всех сотрудников, руководители которых устроились на работу в 2008ом году, но при это сами подчиненные устроились на работу до 2008 года. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```

SELECT
  e."LAST_NAME" AS "Сотрудник",
  e."HIRE_DATE" AS "Дата найма сотрудника",
  m."LAST_NAME" AS "Руководитель",
  m."HIRE_DATE" AS "Дата найма руководителя"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."EMPLOYEES" m
  ON e."MANAGER_ID" = m."EMPLOYEE_ID"
WHERE EXTRACT(YEAR FROM m."HIRE_DATE") = 2008

```

```

    AND EXTRACT(YEAR FROM e."HIRE_DATE") < 2008
ORDER BY m."LAST_NAME", e."LAST_NAME";

```

- Соединяем сотрудников с их руководителями `JOIN`.
- Фильтруем по дате найма: подчинённый до 2008, руководитель — 2008.

```

m."LAST_NAME" AS "руководитель",
m."HIRE_DATE" AS "Дата найма руководителя"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."EMPLOYEES" m
ON e."MANAGER_ID" = m."EMPLOYEE_ID"
WHERE EXTRACT(YEAR FROM m."HIRE_DATE") = 2008
AND EXTRACT(YEAR FROM e."HIRE_DATE") < 2008
ORDER BY m."LAST_NAME", e."LAST_NAME";

```

Showing rows: 1 to 2 of 2
Successfully run. Total query runtime: 235 msec. 2 rows affected.
Total rows: 2 Query complete 00:00:00.235

Рис. 9: Результат выполнения запроса задания 1.9

1.10

Для всех работников, менеджеры которых устроились на работу в январе, и длина названий должностей этих работников(подчиненных) более 15ти символов, сформируйте запрос для вывода названия должности, фамилии работника, даты найма, фамилии руководителя и его даты найма. Результат отсортировать по названию должности, фамилии руководителя, идентификатору работника. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```

SELECT
e."JOB_ID" AS "Должность",
e."LAST_NAME" AS "Сотрудник",
e."HIRE_DATE" AS "Дата найма сотрудника",
m."LAST_NAME" AS "Руководитель",

```

```

m."HIRE_DATE" AS "Дата найма руководителя"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."EMPLOYEES" m
ON e."MANAGER_ID" = m."EMPLOYEE_ID"
WHERE EXTRACT(MONTH FROM m."HIRE_DATE") = 1
AND LENGTH(e."JOB_ID") > 15
ORDER BY e."JOB_ID", m."LAST_NAME", e."EMPLOYEE_ID";

```

- Соединяем сотрудников с их менеджерами `JOIN`.
- Фильтруем менеджеров, нанятых в январе, и подчинённых с длинной должностью > 15 символов.
- Сортировка по должности, фамилии руководителя, ID сотрудника.

```

SELECT m."LAST_NAME" AS "руководитель",
       m."HIRE_DATE" AS "Дата найма руководителя"
  FROM "EmployeesDepartments"."EMPLOYEES" e
 JOIN "EmployeesDepartments"."EMPLOYEES" m
    ON e."MANAGER_ID" = m."EMPLOYEE_ID"
 WHERE EXTRACT(MONTH FROM m."HIRE_DATE") = 1
   AND LENGTH(e."JOB_ID") > 15
 ORDER BY e."JOB_ID", m."LAST_NAME", e."EMPLOYEE_ID";

```

Рис. 10: Результат выполнения запроса задания 1.10

1.11

Напишите запрос для вывода идентификатора отдела и его названия для всех отделов, в которых нет работников. Укажите, какой тип соединения таблиц используется в данном запросе, и задокументируйте результат его выполнения.

```

SELECT
  d."DEPARTMENT_ID",
  d."DEPARTMENT_NAME"
FROM "EmployeesDepartments"."DEPARTMENTS" d
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e

```

```

    ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"
WHERE e."EMPLOYEE_ID" IS NULL
ORDER BY d."DEPARTMENT_ID";

```

- Используется `LEFT JOIN` для включения всех отделов.
- Отбираем только те отделы, где нет сотрудников.
- Сортируем результат по ID отдела.

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer pane, which lists various database objects like Sequences, Tables (with DEPARTMENTS, EMPLOYEES, JOBS, and JOB GRADES), Constraints, Indexes, RLS Policies, Rules, Triggers, JOB_HISTORY, LOCATIONS, Trigger Functions, Types, Views, and countries. The main area is a query editor window titled 'appdb/app@Web PostgreSQL'. It contains a 'Query' tab with the following SQL code:

```

SELECT
    d."DEPARTMENT_ID",
    d."DEPARTMENT_NAME"
FROM "EmployeesDepartments"."DEPARTMENTS" d
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"

```

Below the query is a 'Data Output' tab showing the results of the query execution. The results are as follows:

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	110	Marketing

Messages at the bottom of the Data Output tab indicate: 'Successfully run. Total query runtime: 259 msec. 3 rows affected.' and 'Total rows: 3 Query complete 00:00:00.259'.

Рис. 11: Результат выполнения запроса задания 1.11

Задание 2. Использование групповых функций

2.1

Напишите запрос для вывода идентификатора отдела, количества работников в нём, минимальной, максимальной и средней заработной платы по отделу, а также дат первого и последнего приёма в отдел. Для всех столбцов результата задайте понятные наименования и отсортируйте результат по количеству сотрудников (по убыванию). Задокументируйте результат выполнения запроса.

```

SELECT
    d."DEPARTMENT_ID" AS "ID отдела",
    COUNT(e."EMPLOYEE_ID") AS "Количество сотрудников",
    MIN(e."SALARY") AS "Минимальная зарплата",

```

```

MAX(e."SALARY") AS "Максимальная зарплата",
ROUND(AVG(e."SALARY"), 2) AS "Средняя зарплата",
MIN(e."HIRE_DATE") AS "Дата первого приема",
MAX(e."HIRE_DATE") AS "Дата последнего приема"
FROM "EmployeesDepartments"."DEPARTMENTS" d
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"
GROUP BY d."DEPARTMENT_ID"
ORDER BY "Количество сотрудников" DESC;

```

- Используем `LEFT JOIN` для включения всех отделов.
- Считаем количество сотрудников, минимальные/максимальные/средние зарплаты и даты первого/последнего приема.
- Сортируем результат по количеству сотрудников в отделе по убыванию.

ID отдела	Количество сотрудников	Минимальная зарплата	Максимальная зарплата
1	50	2500	25000
2	80	2500	25000
3	60	2500	25000
4	40	2500	25000
5	30	2500	25000
6	20	2500	25000
7	10	2500	25000
8	5	2500	25000
9	2	2500	25000

Рис. 12: Результат выполнения запроса задания 2.1

2.2

Напишите запрос для вывода названия должности, самого низкого, самого высокого и среднего оклада по ней, а также суммы окладов по каждой должности. Отсортируйте результат по названию должности и задокументируйте результат выполнения запроса.

```

SELECT
    e."JOB_ID" AS "Должность",
    MIN(e."SALARY") AS "Минимальный оклад",
    MAX(e."SALARY") AS "Максимальный оклад",
    ROUND(AVG(e."SALARY"), 2) AS "Средний оклад",
    SUM(e."SALARY") AS "Сумма окладов"
FROM "EmployeesDepartments"."EMPLOYEES" e
GROUP BY e."JOB_ID"
ORDER BY e."JOB_ID";

```

- Используем агрегатные функции для вычисления минимального, максимального, среднего и суммарного оклада по каждой должности.
- Группируем сотрудников по `JOB_ID`.
- Сортировка выполняется по названию должности.

Должность	Минимальный оклад	Максимальный оклад	Средний оклад
AD_PRES	21000.00	21000.00	21000.00
AD_VP	21000.00	21000.00	21000.00
IT_PROG	21000.00	21000.00	21000.00
			Total rows: 8

Рис. 13: Результат выполнения запроса задания 2.2

2.3

Напишите запрос, который позволяет получить список отделов (идентификаторов отделов), их наименований и округленную среднюю заработную плату работников в каждом из них. Для всех столбцов результата задайте понятные наименования, отсортируйте по округленной средней заработной плате и задокументируйте результат выполнения запроса.

```

SELECT
    d."DEPARTMENT_ID" AS "ID отдела",
    d."DEPARTMENT_NAME" AS "Название отдела",
    ROUND(AVG(e."SALARY"), 2) AS "Средняя зарплата"
FROM "EmployeesDepartments"."DEPARTMENTS" d
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"
GROUP BY d."DEPARTMENT_ID", d."DEPARTMENT_NAME"
ORDER BY "Средняя зарплата";

```

- Используем `LEFT JOIN` для включения всех отделов.
- Средняя зарплата вычисляется функцией `AVG` и округляется через `ROUND`.
- Сортируем результат по средней зарплате.

```

SELECT
    d."DEPARTMENT_ID" AS "ID отдела",
    d."DEPARTMENT_NAME" AS "Название отдела",
    ROUND(AVG(e."SALARY"), 2) AS "Средняя зарплата"
FROM "EmployeesDepartments"."DEPARTMENTS" d
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"
GROUP BY d."DEPARTMENT_ID", d."DEPARTMENT_NAME"
ORDER BY "Средняя зарплата";

```

ID отдела	Название отдела	Средняя зарплата
50	Shipping	1212.50
85	Sales	14112.50

Total rows: 9

Рис. 14: Результат выполнения запроса задания 2.3

2.4

Напишите запрос, который позволяет получить список руководителей (их имя, фамилию, должность), у которых количество подчиненных больше 5 и сумма всех зарплат его подчиненных больше 50000. Задокументируйте результат выполнения запроса.

```
SELECT
    m."FIRST_NAME" AS "Имя",
    m."LAST_NAME" AS "Фамилия",
    m."JOB_ID" AS "Должность",
    COUNT(e."EMPLOYEE_ID") AS "Количество подчинённых",
    SUM(e."SALARY") AS "Сумма зарплат подчинённых"
FROM "EmployeesDepartments"."EMPLOYEES" m
JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON e."MANAGER_ID" = m."EMPLOYEE_ID"
GROUP BY m."EMPLOYEE_ID", m."FIRST_NAME", m."LAST_NAME",
m."JOB_ID"
HAVING COUNT(e."EMPLOYEE_ID") > 5
    AND SUM(e."SALARY") > 50000
ORDER BY "Количество подчинённых" DESC;
```

- `JOIN` позволяет сопоставить руководителя с его подчинёнными.
 - Используем `COUNT` и `SUM` для подсчёта и суммы зарплат.
 - `HAVING` фильтрует руководителей по заданным критериям.

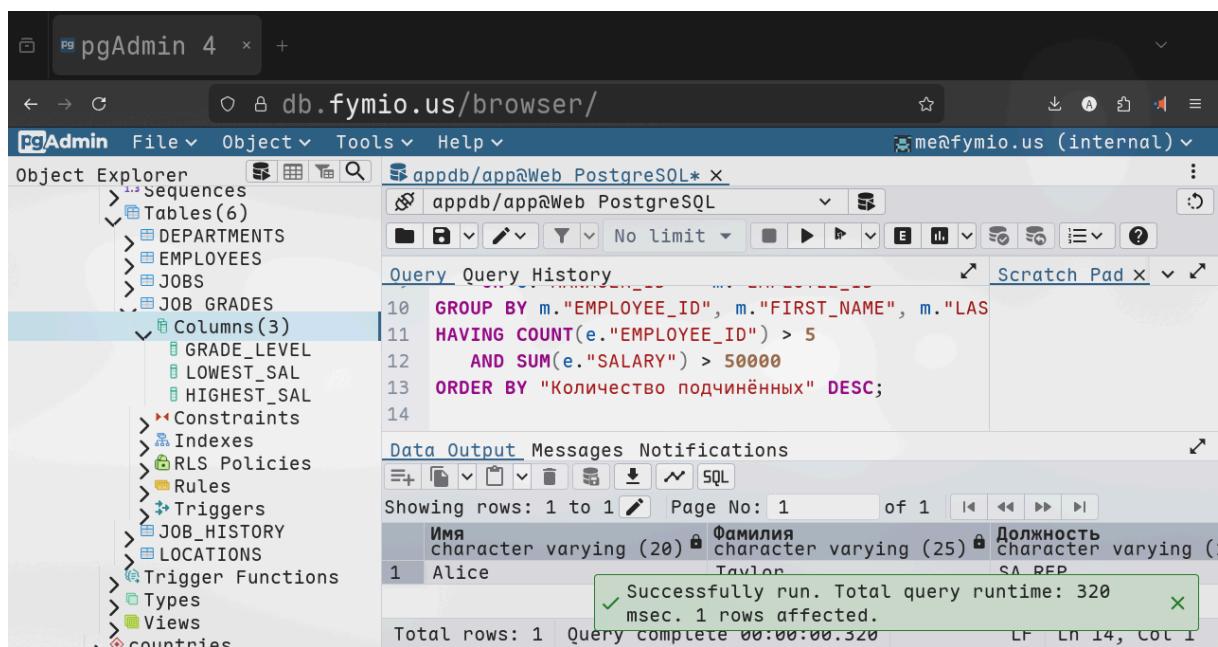


Рис. 15: Результат выполнения запроса задания 2.4

2.5

Напишите запрос для вывода идентификатора отдела и разности между самым высоким и самым низким окладами по каждому отделу. Результат отсортируйте по убыванию разности окладов.

```

SELECT
    e."DEPARTMENT_ID" AS "ID отдела",
    (MAX(e."SALARY") - MIN(e."SALARY")) AS "Разница окладов"
FROM "EmployeesDepartments"."EMPLOYEES" e
GROUP BY e."DEPARTMENT_ID"
ORDER BY "Разница окладов" DESC;

```

- Используем `MAX` и `MIN` для вычисления диапазона зарплат в каждом отделе.
- Группировка по отделу позволяет рассчитать разницу окладов на уровне каждого отдела.
- Сортируем результат по убыванию разницы.

```

pgAdmin 4
db.fymio.us/browser/
File Object Tools Help
me@fymio.us (internal)
Object Explorer Sequences
Tables(6)
DEPARTMENTS EMPLOYEES JOBS JOB GRADES
Columns(3)
GRADE_LEVEL LOWEST_SAL HIGHEST_SAL
Constraints Indexes RLS Policies Rules Triggers
JOB_HISTORY LOCATIONS Trigger Functions Types Views
countries
appdb/app@Web PostgreSQL*
Query History
3   (MAX(e."SALARY") - MIN(e."SALARY")) AS "Разни
4   FROM "EmployeesDepartments"."EMPLOYEES" e
5   GROUP BY e."DEPARTMENT_ID"
6   ORDER BY "Разница окладов" DESC;
7

Data Output Messages Notifications
Showing rows: 1 to 6 Page No: 1 of 1
ID отдела | Разница окладов
smallint | numeric
1 | 90
2 | 20
Total rows: 6 Query complete 00:00:00.335
Successfully run. Total query runtime: 335 msec. 6 rows affected.

```

Рис. 16: Результат выполнения запроса задания 2.5

2.6

Напишите запрос для вывода идентификатора каждого руководителя, имеющего подчинённых, и средней заработной платы этих подчинённых, но только для тех руководителей, которые не получают бонусов, и у которых средняя заработка подчинённых находится в диапазоне от 6000 до 9000. Отсортируйте результат по идентификатору руководителя и задокументируйте результат выполнения запроса.

```
SELECT
```

```
    m."EMPLOYEE_ID" AS "ID руководителя",
    ROUND(AVG(e."SALARY"), 2) AS "Средняя зарплата
    подчинённых"
FROM "EmployeesDepartments"."EMPLOYEES" m
JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON e."MANAGER_ID" = m."EMPLOYEE_ID"
WHERE m."COMMISSION_PCT" IS NULL
GROUP BY m."EMPLOYEE_ID"
HAVING AVG(e."SALARY") BETWEEN 6000 AND 9000
ORDER BY m."EMPLOYEE_ID";
```

- `Self JOIN` позволяет сопоставить руководителя с его подчинёнными.
- Фильтруем руководителей без бонуса и подбираем только тех, у кого средняя зарплата подчинённых в заданном диапазоне.

```
Object Explorer  pgAdmin 4  +  db.fymio.us/browser/
PgAdmin  File  Object  Tools  Help  me@fymio.us (internal)
Object Explorer  Sequences  Tables(6)  DEPARTMENTS  EMPLOYEES  JOBS  JOB GRADES  Columns(3)  Constraints  Indexes  RLS Policies  Rules  Triggers  JOB_HISTORY  LOCATIONS  Trigger Functions  Types  Views  countries
Query  Query History  WHERE m."COMMISSION_PCT" IS NULL
GROUP BY m."EMPLOYEE_ID"
HAVING AVG(e."SALARY") BETWEEN 6000 AND 9000
ORDER BY m."EMPLOYEE_ID";
Data Output  Messages  Notifications
Showing rows: 1 to 3  Page No: 1 of 1
ID руководителя  Средняя зарплата подчинённых
integer  numeric
1  102  8000.00
2  1  10000.00
Total rows: 3  Query complete 00:00:00.221  Successfully run. Total query runtime: 221 msec. 3 rows affected.
```

Рис. 17: Результат выполнения запроса задания 2.6

2.7

Напишите запрос для вывода названия отдела, местоположения отдела (город, адрес) и количества служащих в нём, но только для тех отделов, в которых работники занимают различные должности. Для всех столбцов результата задайте понятные наименования и отсортируйте результат по количеству

служащих (по убыванию). Результат выполнения запроса задокументируйте.

```
SELECT
    d."DEPARTMENT_NAME" AS "Отдел",
    l."CITY" || ', ' || l."STREET_ADDRESS" AS
    "Местоположение",
    COUNT(e."EMPLOYEE_ID") AS "Количество сотрудников"
FROM "EmployeesDepartments"."DEPARTMENTS" d
JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"
JOIN "EmployeesDepartments"."LOCATIONS" l
    ON d."LOCATION_ID" = l."LOCATION_ID"
GROUP BY d."DEPARTMENT_NAME", l."CITY", l."STREET_ADDRESS",
d."DEPARTMENT_ID"
HAVING COUNT(DISTINCT e."JOB_ID") > 1
ORDER BY "Количество сотрудников" DESC;
```

- Соединяем отделы с сотрудниками и местоположениями **INNER JOIN**.
 - Считаем количество сотрудников и проверяем, что в отделе есть различные должности.

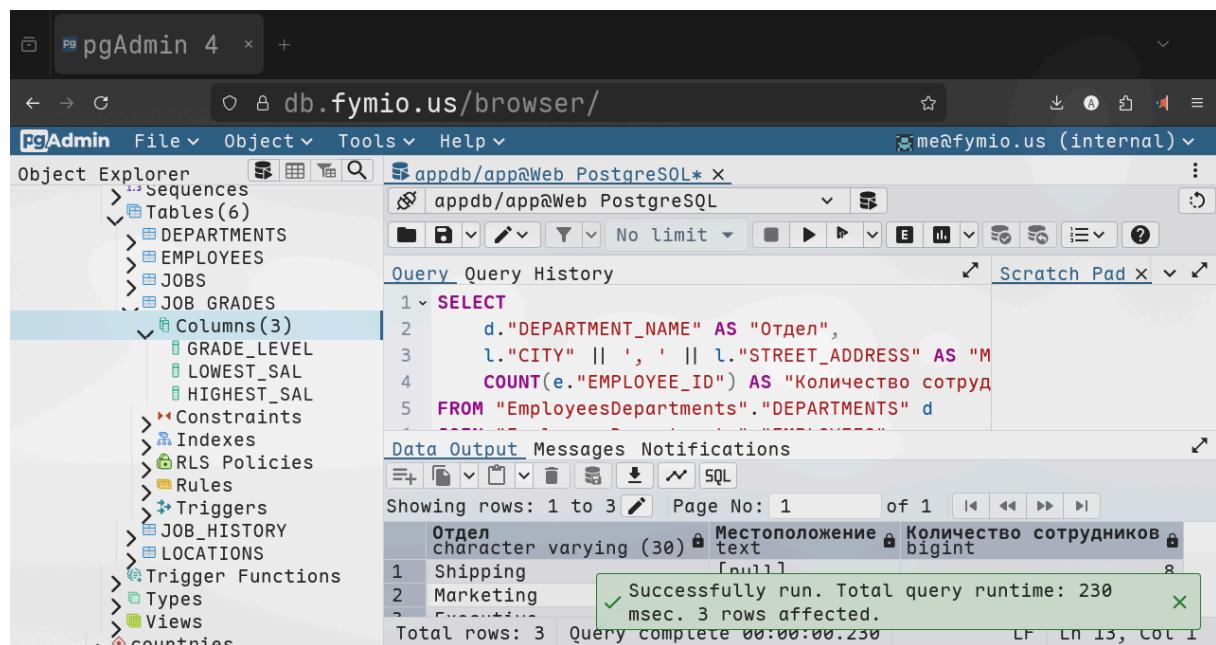


Рис. 18: Результат выполнения запроса задания 2.7

2.8

Напишите запрос для вывода года и количества принятых на работу сотрудников в указанном году по всем годам. Результат отсортируовать по количеству принятых на работу сотрудников в год и задокументировать.

```
SELECT
    EXTRACT(YEAR FROM e."HIRE_DATE") AS "Год",
    COUNT(e."EMPLOYEE_ID") AS "Количество принятых"
FROM "EmployeesDepartments"."EMPLOYEES" e
GROUP BY EXTRACT(YEAR FROM e."HIRE_DATE")
ORDER BY "Количество принятых" DESC;
```

- Извлекаем год найма с помощью `EXTRACT(YEAR ...)`.
- Считаем количество сотрудников, принятых в каждый год `COUNT`.
- Сортируем по количеству принятых сотрудников по убыванию.

```
3     COUNT(e."EMPLOYEE_ID") AS "Количество принятых"
4   FROM "EmployeesDepartments"."EMPLOYEES" e
5 GROUP BY EXTRACT(YEAR FROM e."HIRE_DATE")
6 ORDER BY "Количество принятых" DESC;
7
```

Год	Количество принятых
2012	6
2014	10

Successfully run. Total query runtime: 171 msec. 10 rows affected.

Рис. 19: Результат выполнения запроса задания 2.8

2.9

Напишите запрос, который выводит длину имени и количество сотрудников с соответствующей длиной имени. В результат включите только тех сотрудников, у которых длина имени больше 5, а количество сотрудников с такой длиной – больше 3. Результат отсортируйте по длине имени и задокументируйте.

```

SELECT
    LENGTH(e."FIRST_NAME") AS "Длина имени",
    COUNT(e."EMPLOYEE_ID") AS "Количество сотрудников"
FROM "EmployeesDepartments"."EMPLOYEES" e
WHERE LENGTH(e."FIRST_NAME") > 5
GROUP BY LENGTH(e."FIRST_NAME")
HAVING COUNT(e."EMPLOYEE_ID") > 3
ORDER BY "Длина имени";

```

- Считаем количество сотрудников по длине имени `LENGTH + COUNT`.
- Отбираем только имена длиннее 5 символов и длины, встречающиеся у более чем 3 сотрудников.
- Сортируем результат по длине имени.

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer tree shows the database structure with tables like DEPARTMENTS, EMPLOYEES, JOBS, and JOB GRADES. The current tab is the Query window, which contains the following SQL code:

```

SELECT
    LENGTH(e."FIRST_NAME") AS "Длина имени",
    COUNT(e."EMPLOYEE_ID") AS "Количество сотрудников"
FROM "EmployeesDepartments"."EMPLOYEES" e
WHERE LENGTH(e."FIRST_NAME") > 5
GROUP BY LENGTH(e."FIRST_NAME")
HAVING COUNT(e."EMPLOYEE_ID") > 3
ORDER BY "Длина имени";

```

Below the code, the Data Output pane shows the results of the query execution:

- Длина имени integer
- Successfully run. Total query runtime: 177 msec. 0 rows affected.
- Total rows: 0 Query complete 00:00:00.177

Рис. 20: Результат выполнения запроса задания 2.9

2.10

Напишите запрос, который выводит названия отделов, их идентификационный номер, адрес и город, а также количество работников в каждом отделе, включая те, где пока нет ни одного работника. Укажите, какой тип соединения таблиц используется в данном запросе. Для всех столбцов результата задайте понятные наименования, отсортируйте по номеру отдела и задокументируйте.

```

SELECT
    d."DEPARTMENT_NAME" AS "Название отдела",
    d."DEPARTMENT_ID" AS "ID отдела",
    l."STREET_ADDRESS" AS "Адрес",
    l."CITY" AS "Город",
    COUNT(e."EMPLOYEE_ID") AS "Количество сотрудников"
FROM "EmployeesDepartments"."DEPARTMENTS" d
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"
JOIN "EmployeesDepartments"."LOCATIONS" l
    ON d."LOCATION_ID" = l."LOCATION_ID"
GROUP BY d."DEPARTMENT_ID", d."DEPARTMENT_NAME",
l."STREET_ADDRESS", l."CITY"
ORDER BY d."DEPARTMENT_ID";

```

- Считаем количество сотрудников в каждом отделе, включая отделы без сотрудников `LEFT JOIN`.
- Получаем адрес и город через соединение с `LOCATIONS`.
- Результат отсортирован по номеру отдела.

The screenshot shows the pgAdmin 4 interface with a query window containing the following SQL code:

```

SELECT
    d."DEPARTMENT_NAME" AS "Название отдела",
    d."DEPARTMENT_ID" AS "ID отдела",
    l."STREET_ADDRESS" AS "Адрес",
    l."CITY" AS "Город",
    COUNT(e."EMPLOYEE_ID") AS "Количество сотрудников"
FROM "EmployeesDepartments"."DEPARTMENTS" d
LEFT JOIN "EmployeesDepartments"."EMPLOYEES" e
    ON d."DEPARTMENT_ID" = e."DEPARTMENT_ID"
JOIN "EmployeesDepartments"."LOCATIONS" l
    ON d."LOCATION_ID" = l."LOCATION_ID"
GROUP BY d."DEPARTMENT_ID", d."DEPARTMENT_NAME",
l."STREET_ADDRESS", l."CITY"
ORDER BY d."DEPARTMENT_ID";

```

The results pane displays the following data:

Название отдела	ID отдела	Адрес	Город
Administration	1	character varying (30)	smallint
Marketing	2	character varying (40)	character varying (40)

Total rows: 2 Query complete 00:00:00.207

Рис. 21: Результат выполнения запроса задания 2.10

2.11

Напишите запрос, который выводит название должности, количество работников, занимающих эту должность, а также среднюю заработную плату по каждой должности в отделах

Administration и IT. В результат включите только те должности, где средняя зарплата превышает 4000, и на которых работает более двух сотрудников. Для всех столбцов результата задайте понятные наименования, отсортируйте данные по убыванию количества сотрудников и задокументируйте.

```
SELECT
    e."JOB_ID" AS "Должность",
    COUNT(e."EMPLOYEE_ID") AS "Количество сотрудников",
    ROUND(AVG(e."SALARY"), 2) AS "Средняя зарплата"
FROM "EmployeesDepartments"."EMPLOYEES" e
JOIN "EmployeesDepartments"."DEPARTMENTS" d
    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
WHERE d."DEPARTMENT_NAME" IN ('Administration', 'IT')
GROUP BY e."JOB_ID"
HAVING COUNT(e."EMPLOYEE_ID") > 2
    AND AVG(e."SALARY") > 4000
ORDER BY "Количество сотрудников" DESC;
```

- Соединяем таблицы сотрудников и отделов.
- Фильтруем только нужные отделы.
- Группируем по должности и считаем количество сотрудников и среднюю зарплату.
- Фильтруем по условиям `HAVING` и сортируем результат по количеству сотрудников.

```

    ON e."DEPARTMENT_ID" = d."DEPARTMENT_ID"
WHERE d."DEPARTMENT_NAME" IN ('Administration',
GROUP BY e."JOB_ID"
HAVING COUNT(e."EMPLOYEE_ID") > 2
AND AVG(e."SALARY") > 4000
ORDER BY "Количество сотрудников" DESC;

```

Должность | Количество сотрудников | Среднее зарплата
1 TT PRG | 247 | Successfully run. Total query runtime: 247 msec. 1 rows affected.
Total rows: 1 Query complete 00:00:00.247

Рис. 22: Результат выполнения запроса задания 2.11

Выводы и анализ результатов работы

В ходе выполнения работы удалось последовательно отработать ключевые навыки работы с реляционными базами данных. Основной целью было научиться применять различные виды соединений таблиц, фильтрацию, группировку, агрегирование и оформление результатов. Все эти задачи были выполнены: мы строили запросы с `INNER JOIN`, исследовали структуру связанных таблиц, получали данные о сотрудниках и отделах, вычисляли агрегаты и формировали отчёты, соответствующие условиям заданий.

Удалось добиться полного понимания, как извлекать данные из нескольких таблиц и объединять их в единую осмысленную выборку. Отдельные задания требовали работы с условиями, правильного выбора типа соединения и обращения к именованным полям.

В итоге была закреплена практика использования агрегатных функций, фильтрации по шаблону, сортировки результатов. Работа показала, как шаг за шагом строится аналитика поверх обычных данных, и позволила сформировать цельное понимание того, как sql-запросы применяются для анализа и обработки информации в реальных базах данных.