Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ по лабораторной работе №**12

по дисциплине «Базы данных»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Сафонов

подпись

**Проверил:**

Кандидат тех. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

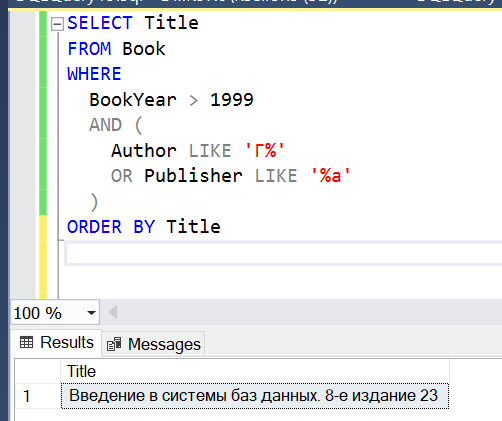
подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

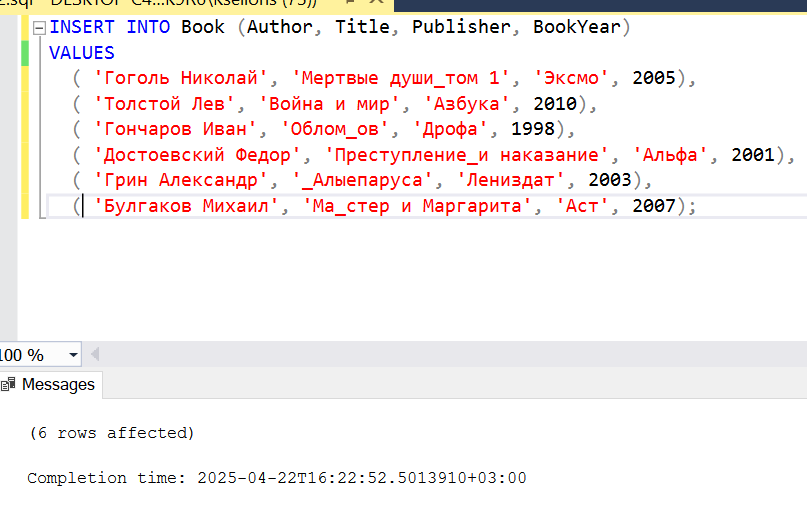
Санкт-Петербург, 2025

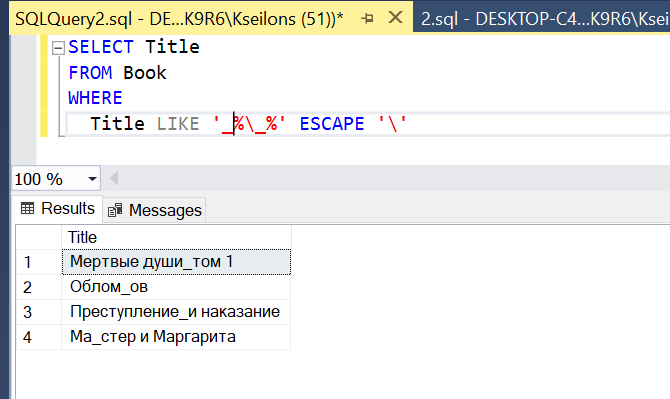
# ЛАБОРАТОРНАЯ РАБОТА № 1 (SQL SERVER) ОДИННАДЦАТОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ».

Задание. Напишите на SQL запрос, который выведет в обратном алфавитном порядке названия книг, выпущенных после 1999 года, фамилия автора (значение поля Author) у которых начинается с буквы «Г» или название издательства заканчивается буквой «а».

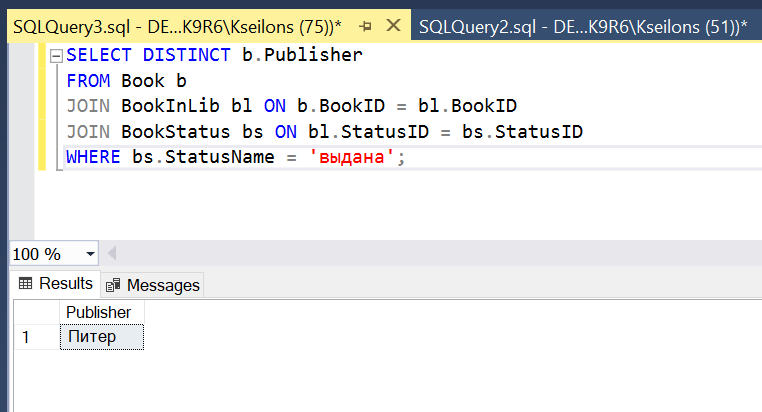


Задание. Напишите запрос, выводящий названия книг, содержащие символ подчеркивания во второй или любой следующей позиции. (Первый символ может быть произвольным, но если это подчеркивание, и оно единственное, такое название выводить не надо).

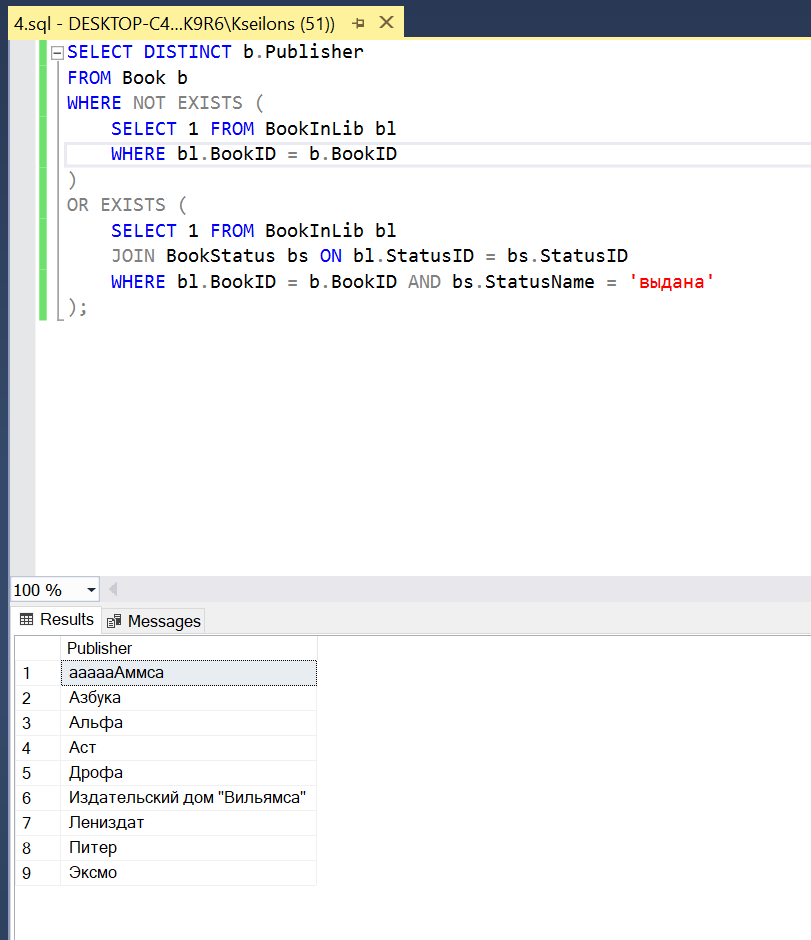




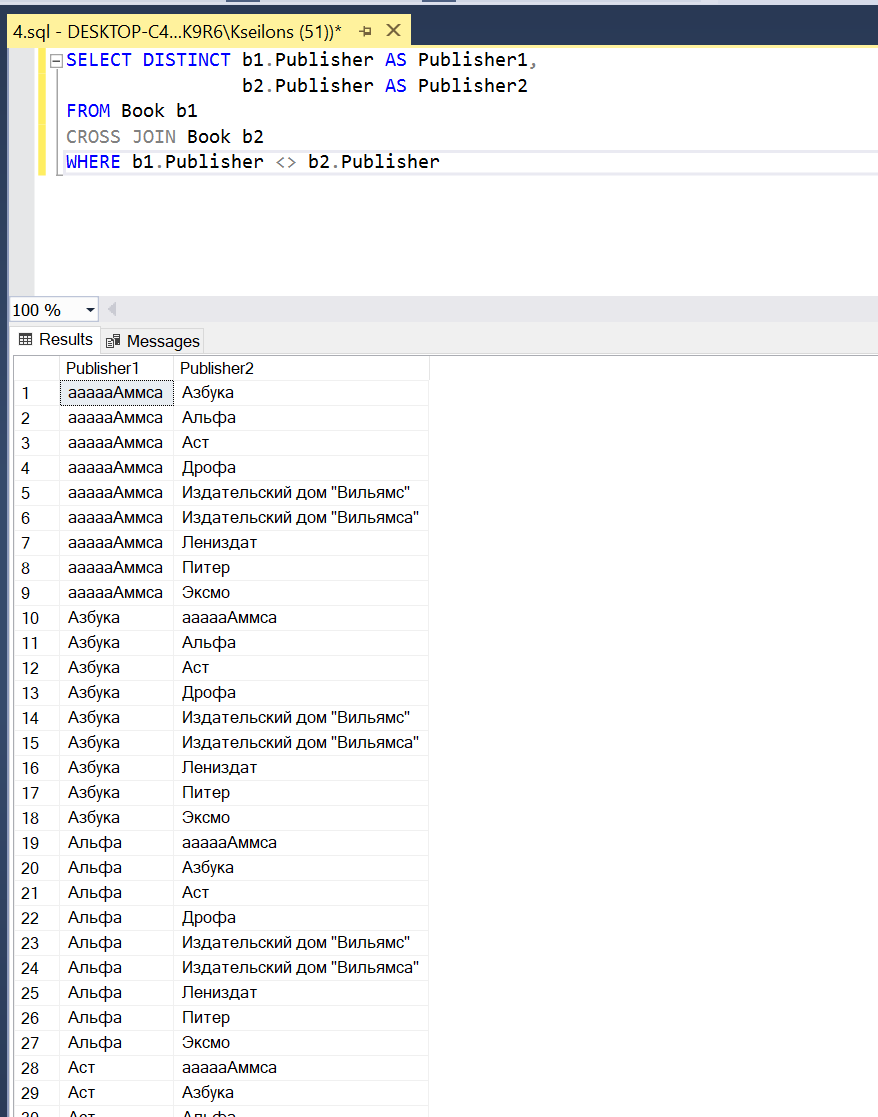
Задание. Напишите запрос, выводящий без повторения список издательств, таких, что хотя бы одна книга этого издательства имеет статус «выдана».



Задание. Измените предыдущий запрос, чтобы он выводил без повторения список издательств, таких, что или ни одного экземпляра хотя бы одной книги этого издательства нет в библиотеке (в таблице BookInLib), или хотя бы одна книга этого издательства имеет статус «выдана»



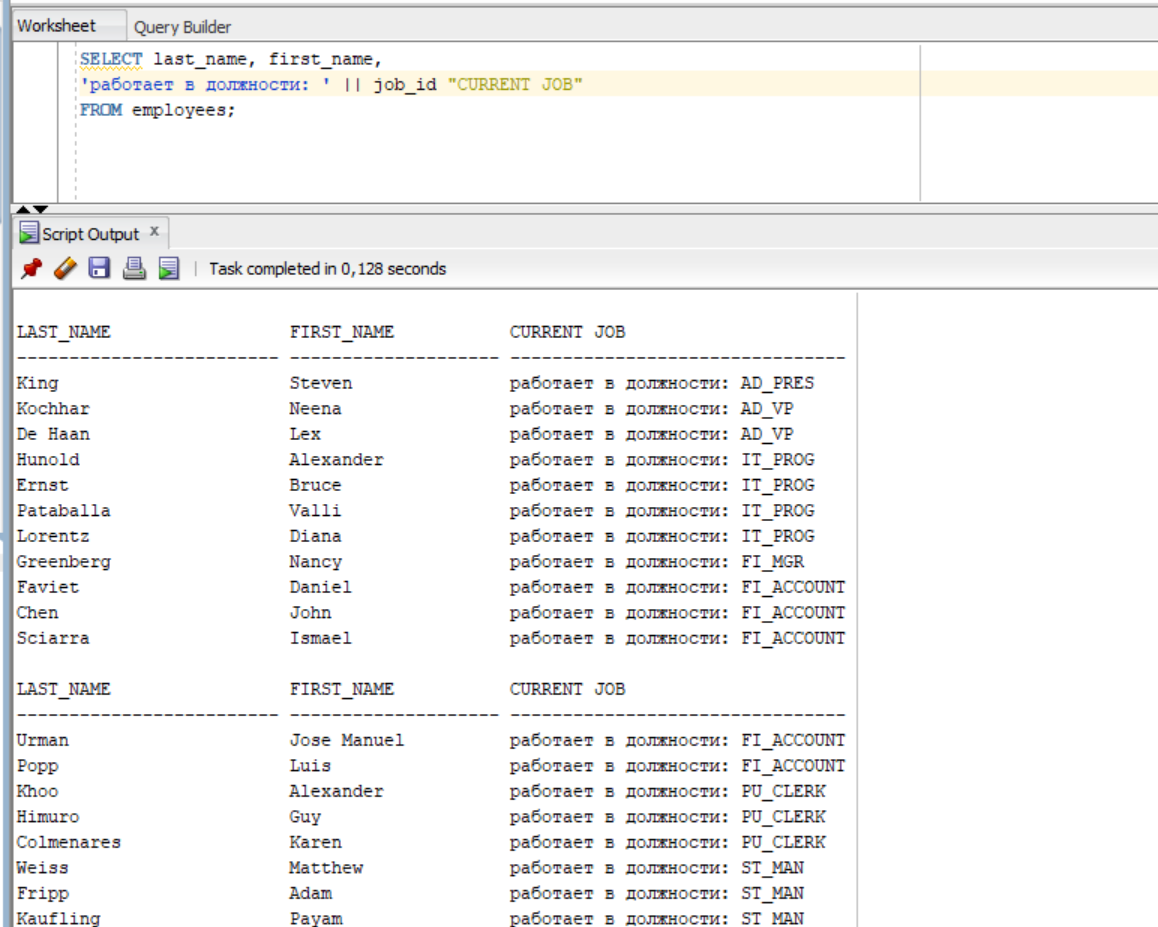
Задание. По аналогии с рассмотренным примером напишите свой запрос, выводящий без повторения все сочетания разных издательств, упомянутых в таблице Book



# ЛАБОРАТОРНАЯ РАБОТА (Oracle) ОДИННАДЦАТОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ». ВЫБОРКА ДАННЫХ: ПРОСТЫЕ ЗАПРОСЫ, ГРУППИРОВКА, СОЕДИНЕНИЯ ТАБЛИЦ

Команда ниже содержит 4 ошибки, найдите и исправьте их.

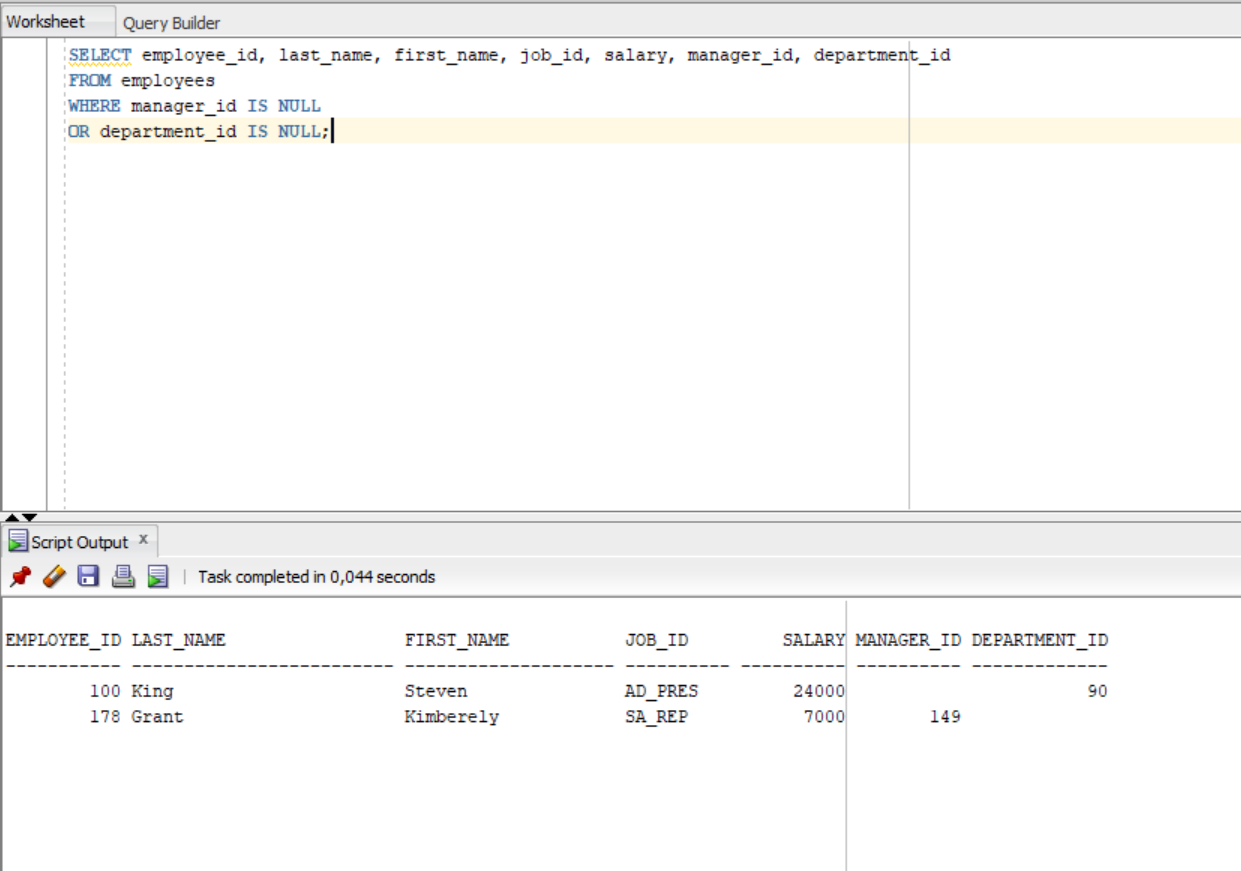
SELECT last\_name, first\_name 'работает в должности: ' + jobid CURRENT JOB FROM employees;



Создайте запрос для получения в одном отчёте номера, фамилии, имени, должности, зарплаты, номера руководителя и номера отдела для:

a. сотрудников высшего звена, т.е. таких, которые не имеют руководителя (менеджера) и

b. сотрудников, не приписанных ни к одному отделу

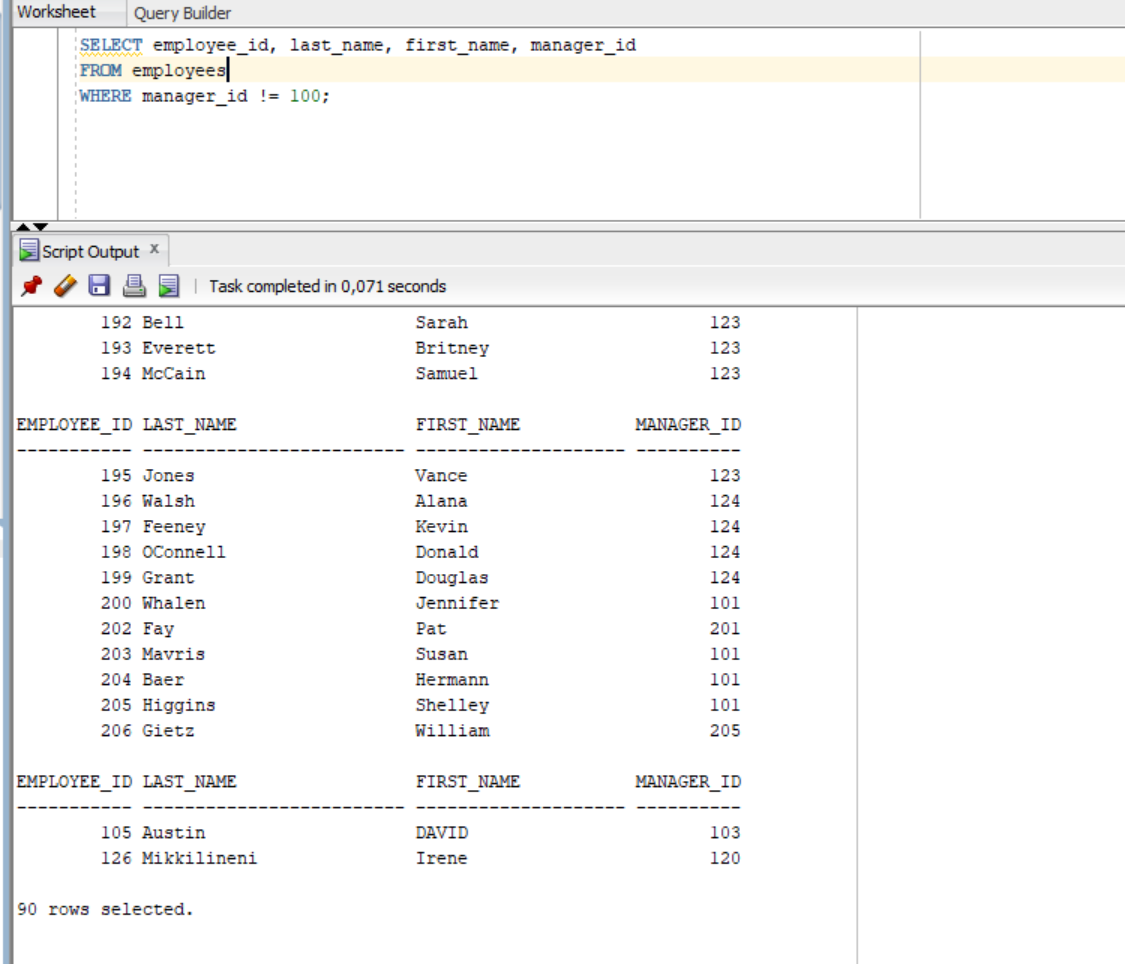


Вы хотите узнать, какие сотрудники подчиняются любым руководителям, кроме сотрудника высшего звена (с номером 100) и выполняете следующий запрос:

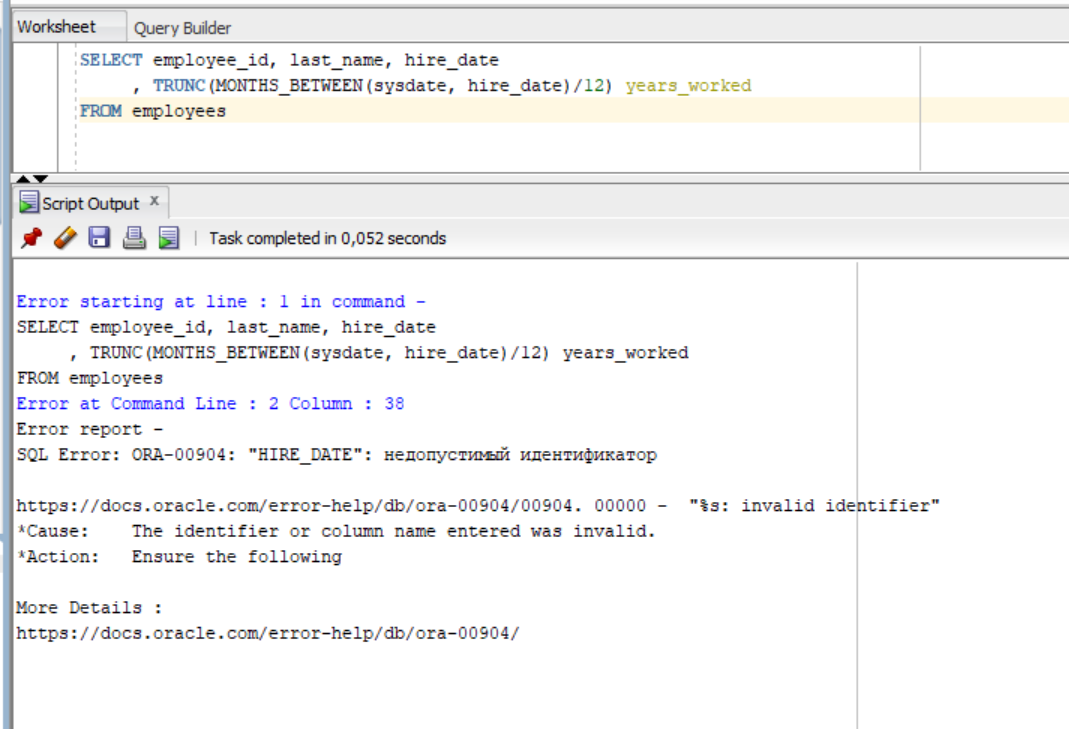
SELECT employee\_id, last\_name, first\_name, manager\_id FROM employees WHERE manager\_id NOT IN (100, null);

При просмотре таблицы «вручную» вы можете увидеть, что такие сотрудники существуют (например, это сотрудники с номерами 103 и 104), но в результат запроса они не попали.

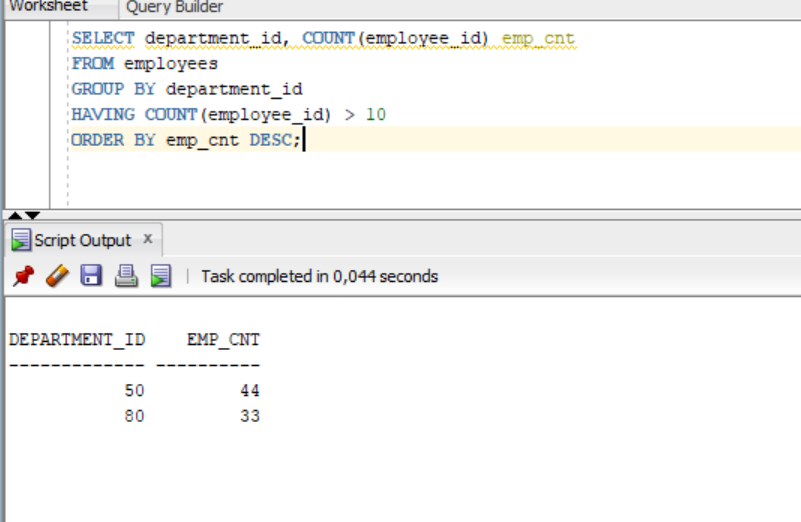
Как вы думаете, почему? Как следует переписать запрос, чтобы получить верный ответ?



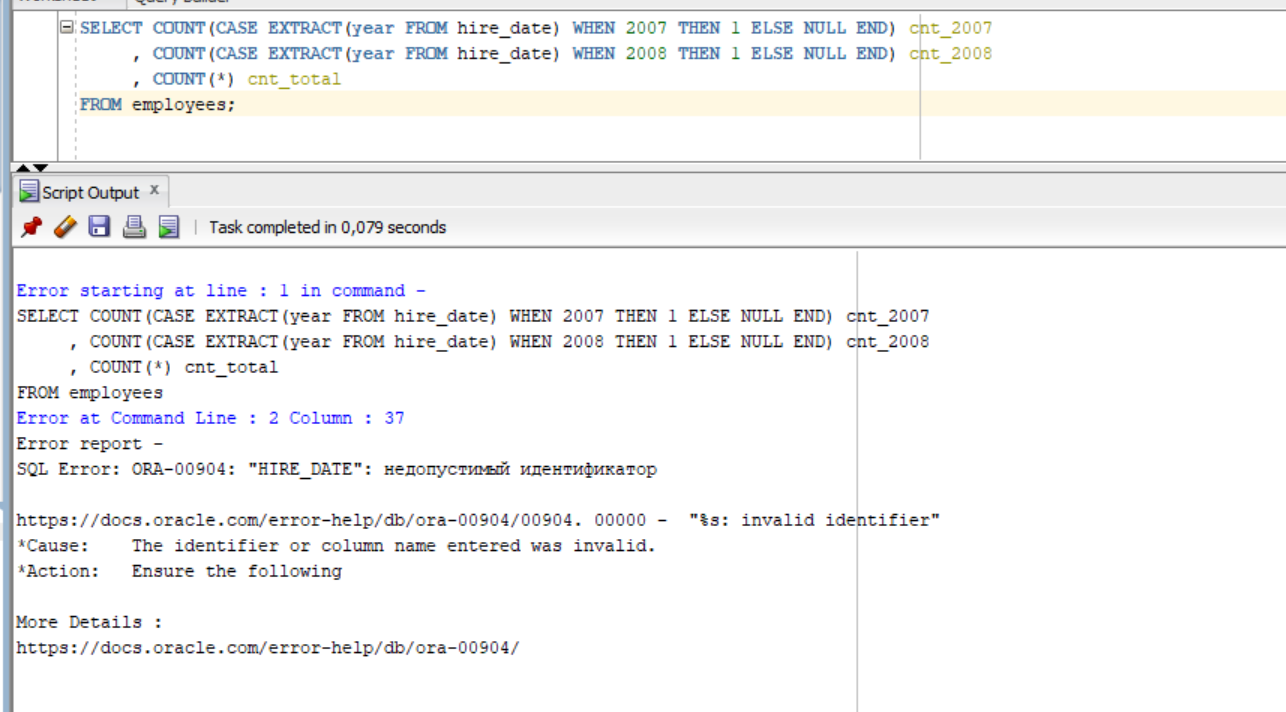
Сформируйте запрос для определения полного числа лет, отработанных каждым сотрудником. Назовите столбец YEARS\_WORKED. Выведите дополнительно номер сотрудника, его фамилию и дату приёма на работу. Отсортируйте результат в порядке возрастания числа проработанных лет. Если какие-то сотрудники трудятся в компании одинаковое количество лет, отсортируйте их по фамилиям (по алфавиту).



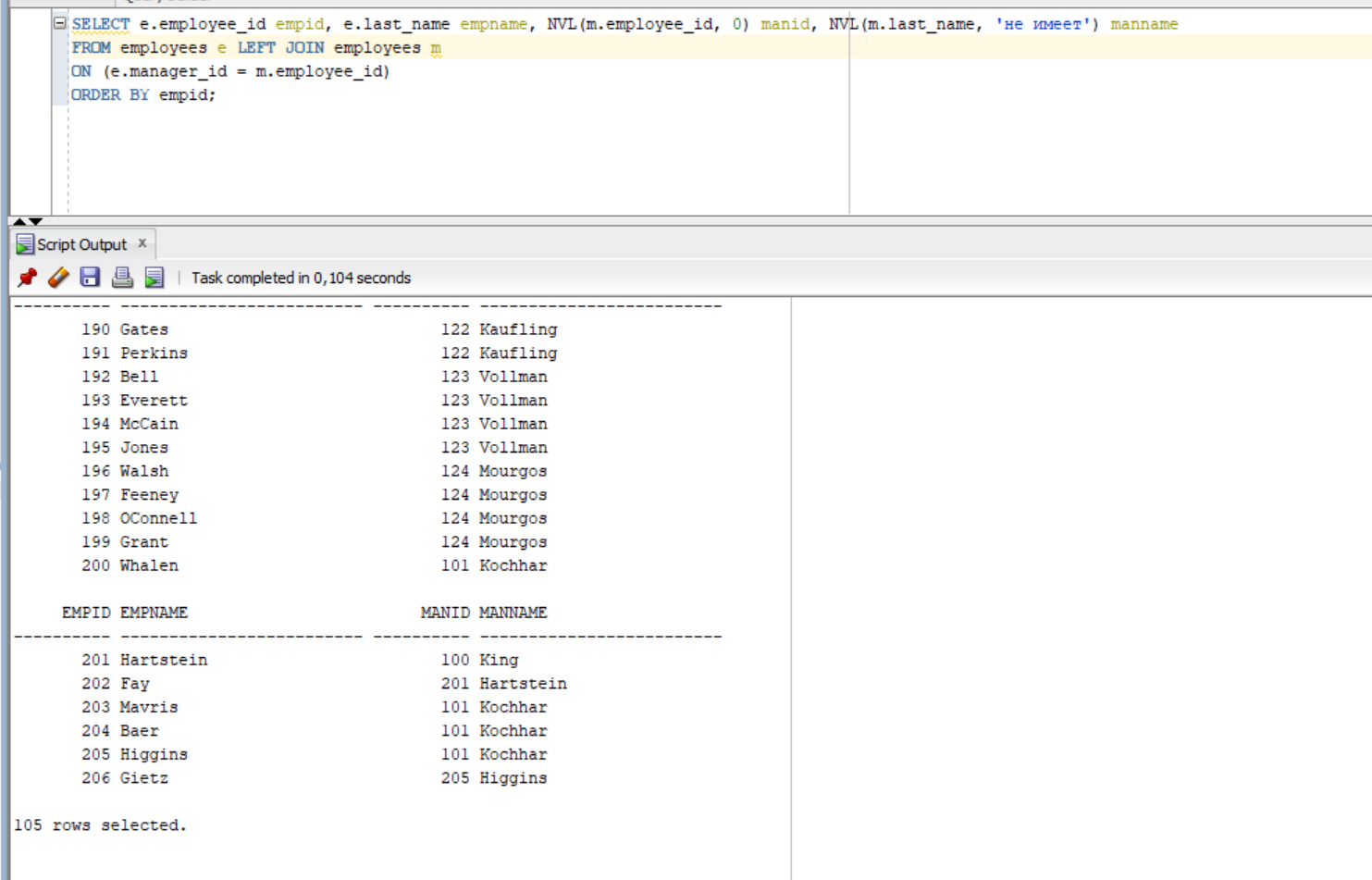
Сформируйте запрос, позволяющий определить, в каких отделах трудится более 10 сотрудников. Выведите номер отдела и общее количество сотрудников. Отсортируйте результат по уменьшению общего количества сотрудников.



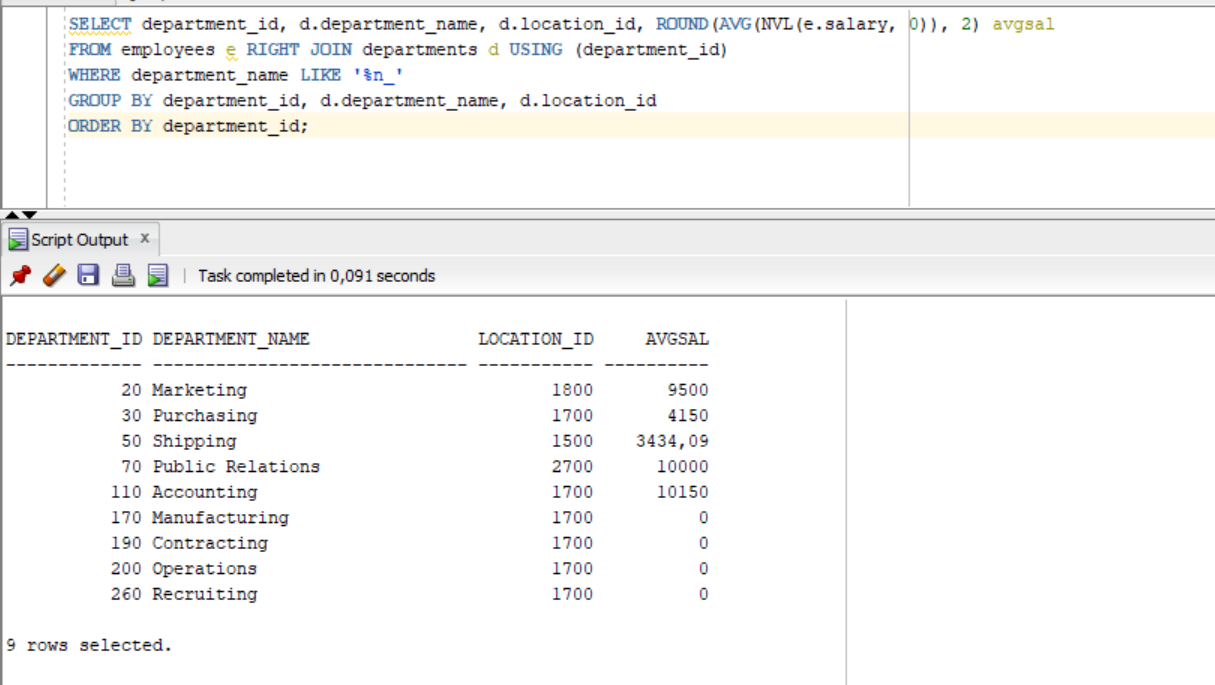
Сформируйте запрос для определения количества сотрудников, нанятых в 2007 и 2008 годах и общего количества сотрудников.



Сформируйте запрос для вывода фамилий и номеров абсолютно всех служащих вместе с фамилиями и номерами их менеджеров. Присвойте соответствующие псевдонимы столбцам. Для сотрудника высшего звена в столбцах с номером и фамилией руководителя выведите 0 и 'не имеет' соответственно. Отсортируйте результат по возрастанию номера сотрудника.

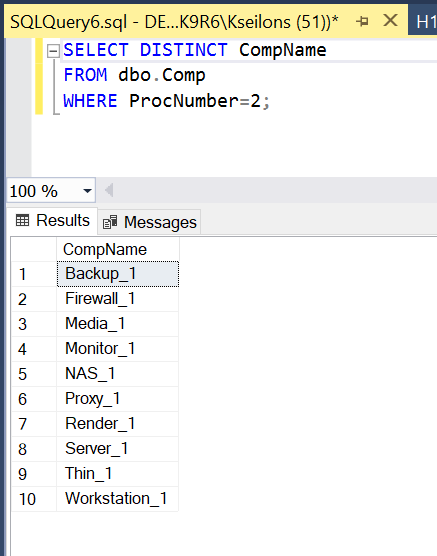


Сформируйте запрос для вывода сводной информации по отделам, предпоследняя буква в названии которых – n. Включите в результат следующие столбцы: номер, наименование, местоположение отдела, а также средний оклад работающих в нём сотрудников, округлённый до 2 цифр после запятой. Обеспечьте вывод отделов, в которых нет сотрудников. Средний оклад в таких отделах должен быть равен нулю. Отсортируйте результат по возрастанию номера отдела.

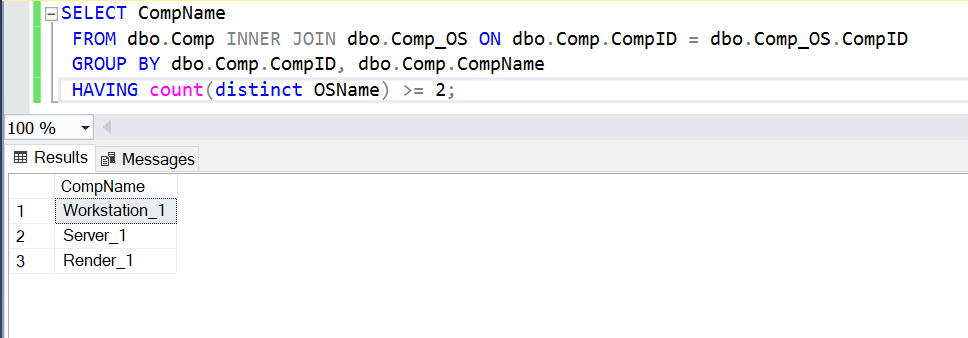


# ЛАБОРАТОРНАЯ РАБОТА № 1 (SQL SERVER) ДВЕНАДЦАТОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ»

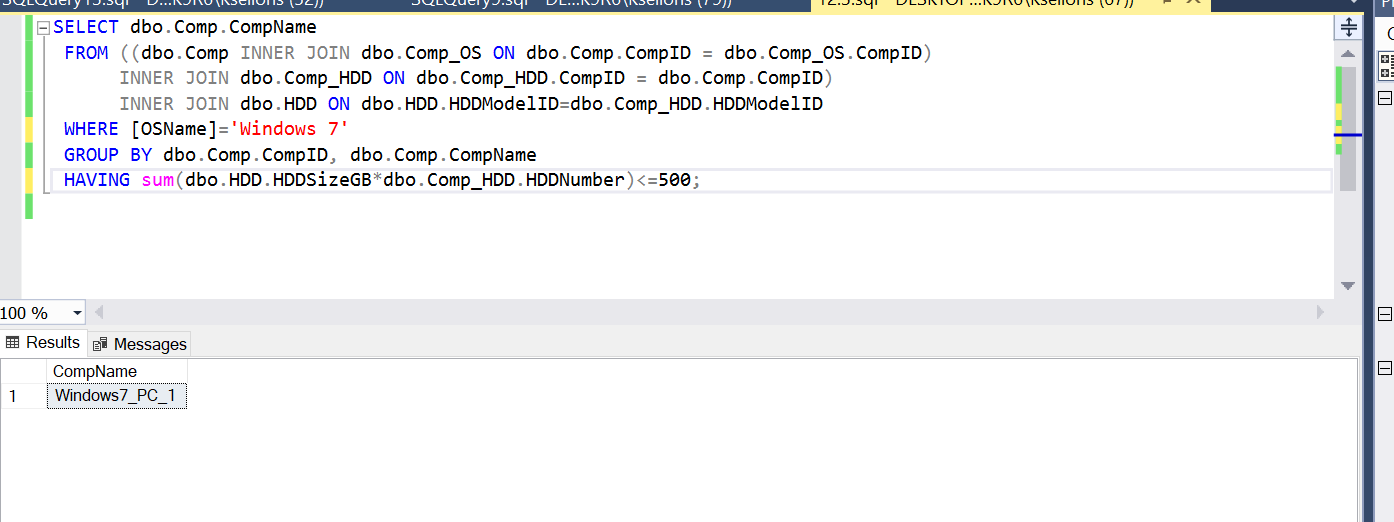
1. Список без повторения компьютеров с двумя процессорами.



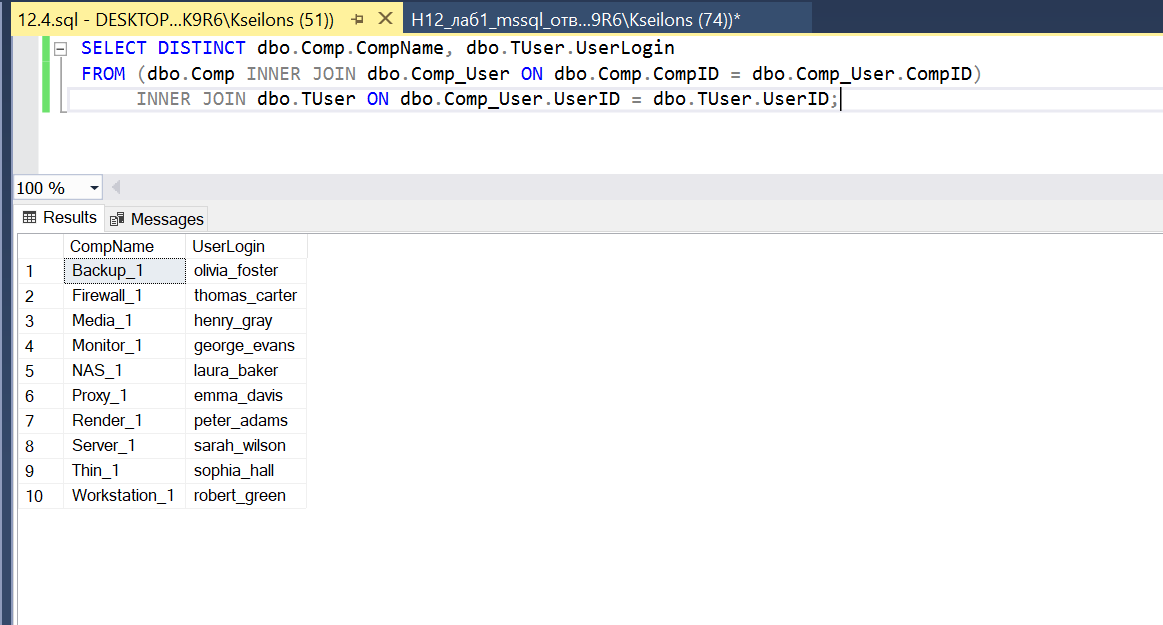
1. Список без повторения компьютеров, на которых установлены две и более различные операционные системы.



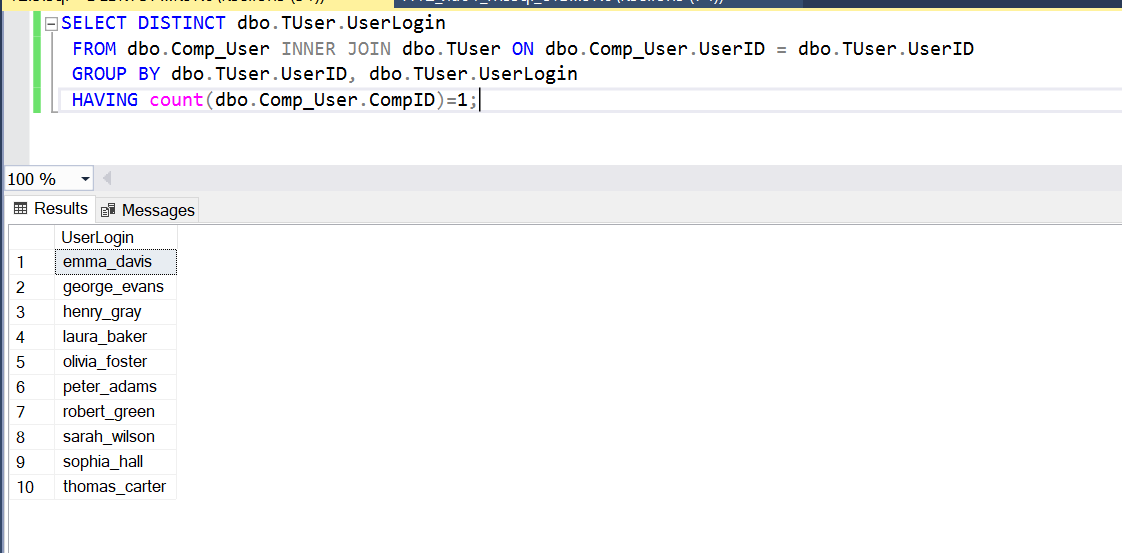
1. Список компьютеров, таких, что на компьютере ОС Windows 7 и суммарный объем жестких дисков компьютера не превышает 500 Гб (у компьютера может быть более 1 диска).



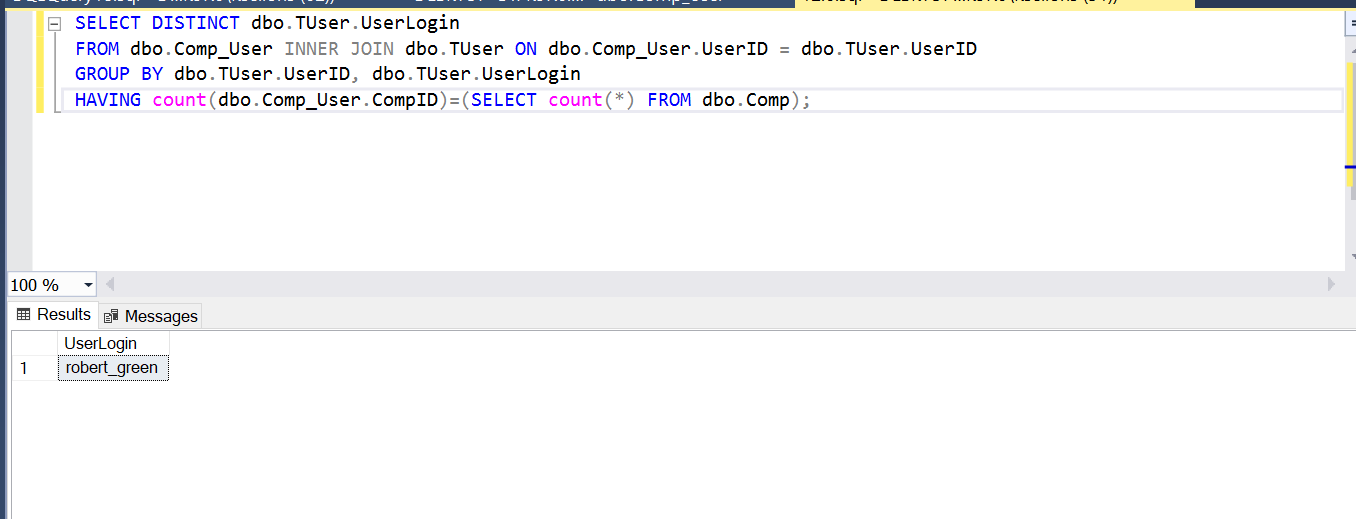
1. Список компьютеров и пользователей, работающих за ними.



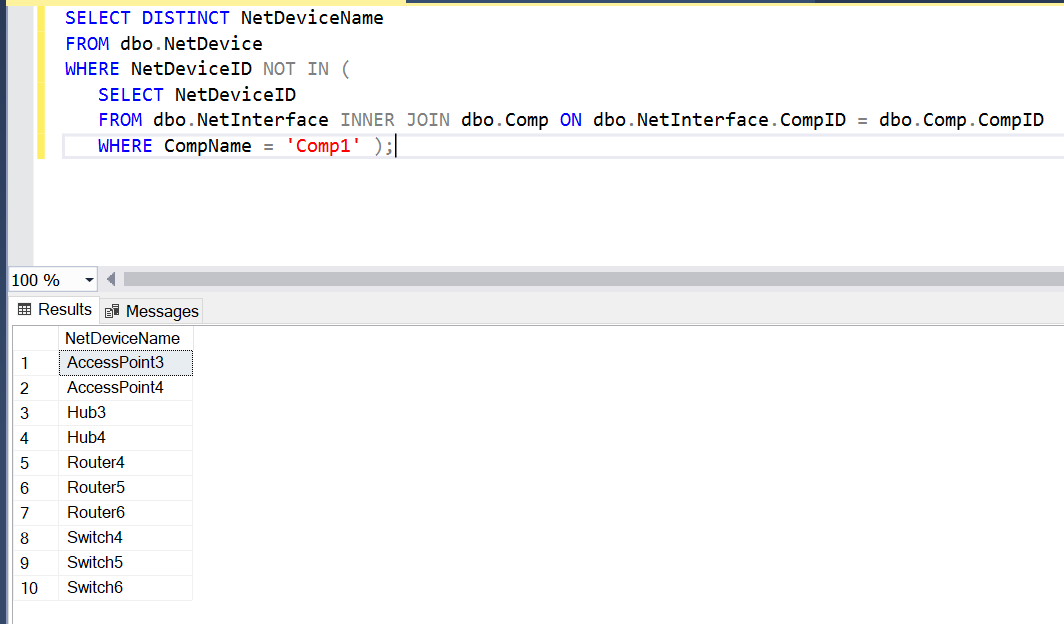
1. Список без повторения пользователей, работающих только за одним компьютером.



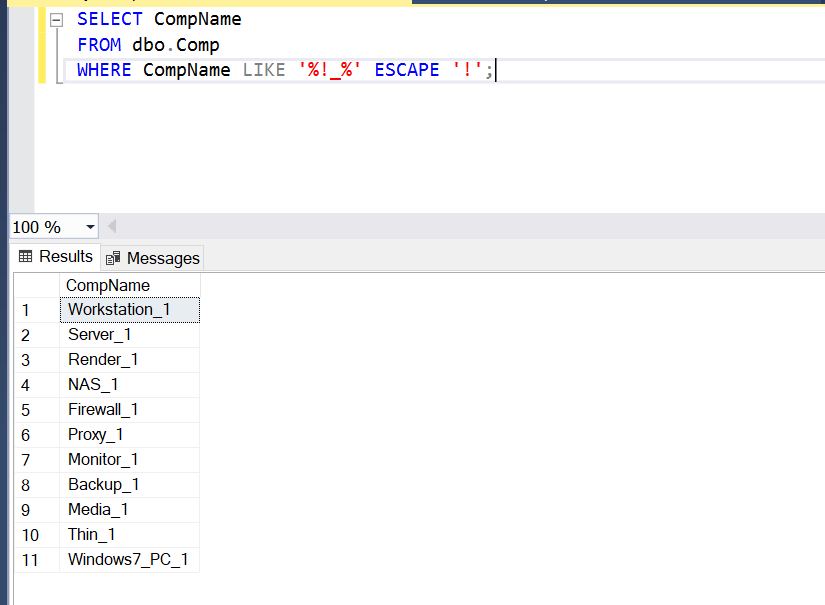
1. Список без повторения пользователей, работающих за всеми компьютерами организации.



1. Список без повторения сетевых устройств (идентификатор и название), к которым непосредственно не подключен компьютер с названием Comp1.

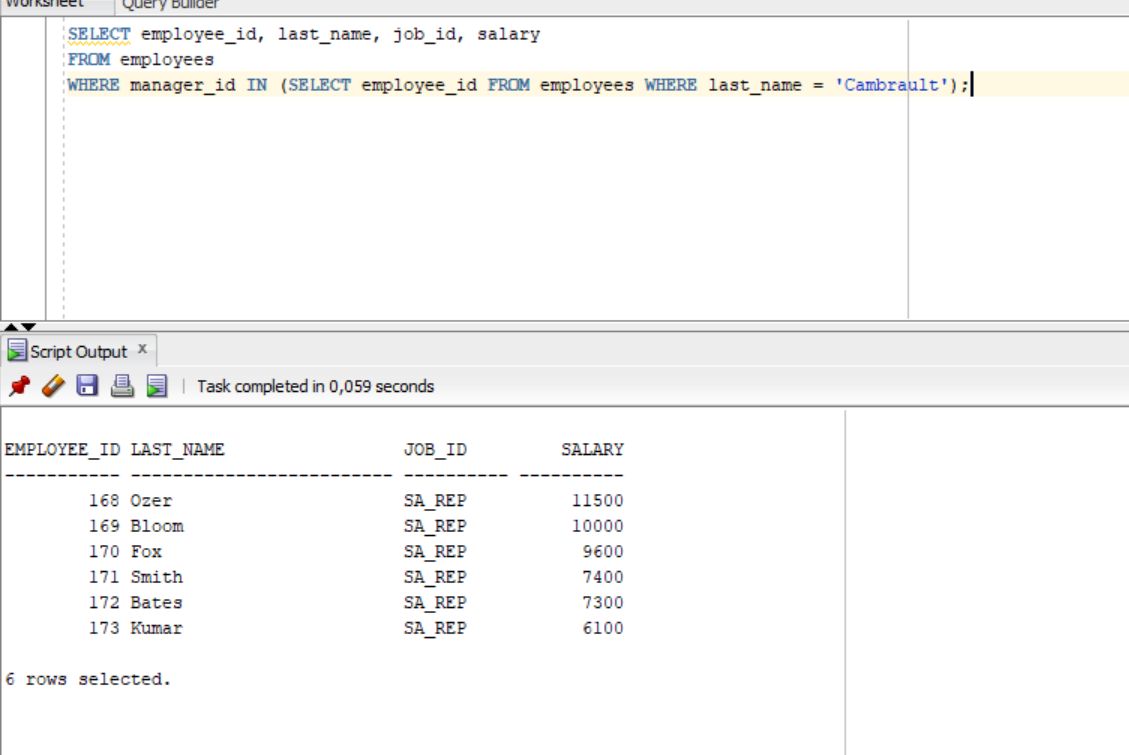


8. Список компьютеров, в именах которых встречается символ подчеркивания «\_».

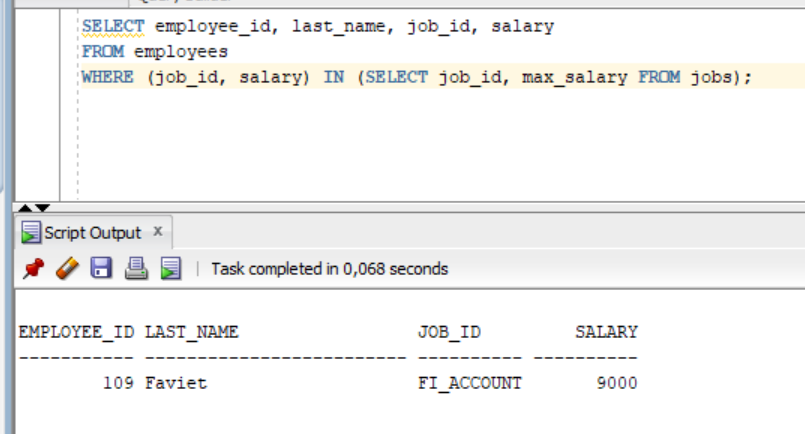


# ЛАБОРАТОРНАЯ РАБОТА (Oracle) ДВЕНАДЦАТОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ». ВЫБОРКА ДАННЫХ: ПОДЗАПРОСЫ, ОПЕРАТОРЫ ДЛЯ РАБОТЫ СО МНОЖЕСТВАМИ ЗАПИСЕЙ

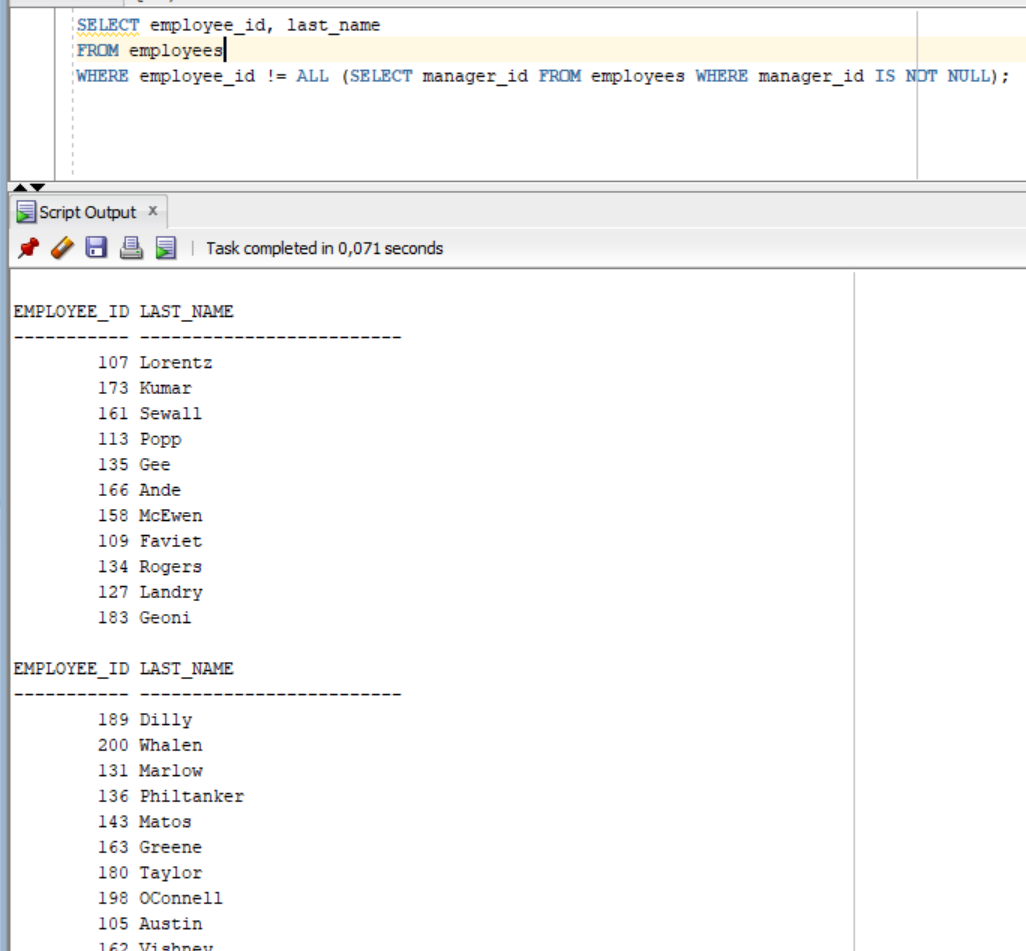
Сформируйте запрос для определения номеров, фамилий, идентификаторов должности и окладов сотрудников, руководитель которых носит фамилию 'Cambrault'



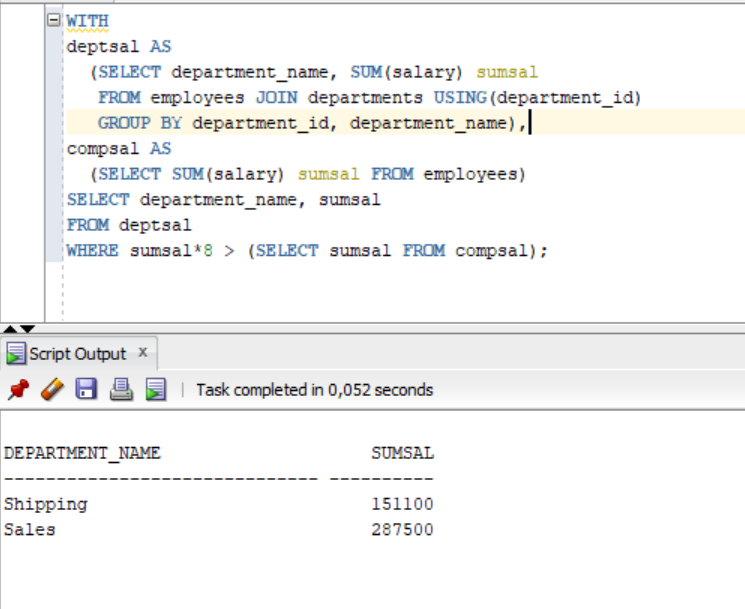
Используя многостолбцовый подзапрос, сформируйте запрос, возвращающий номера, фамилии, идентификаторы должности и оклад для тех сотрудников, чей оклад совпадает с максимальным по их должности.



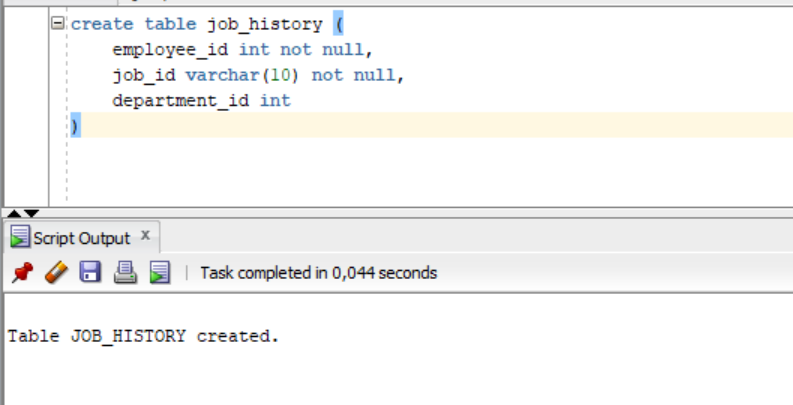
Создайте подзапрос для получения номеров и фамилий сотрудников, не являющихся руководителями других сотрудников. Используйте оператор ALL для сравнения с результатами подзапроса.

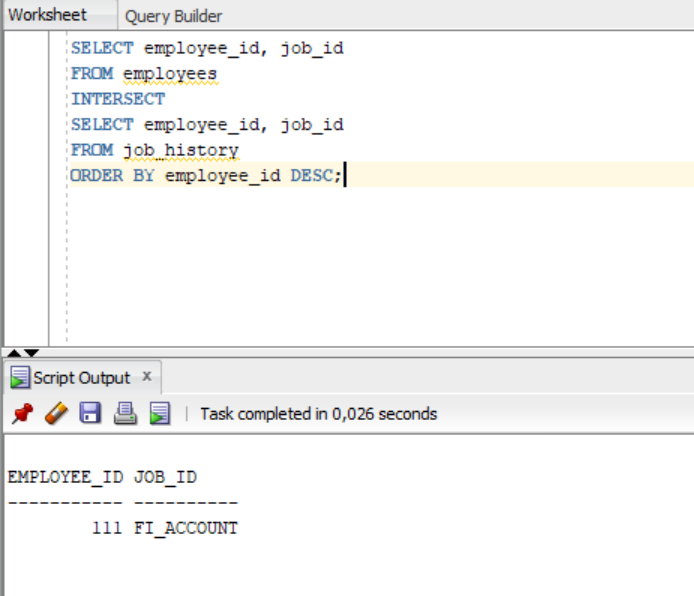


Используйте раздел WITH при формировании запроса для определения наименования и фондов заработной платы (суммы окладов всех сотрудников) таких отделов, размер фондов которых больше 1/8 от общего фонда заработной платы всей компании.

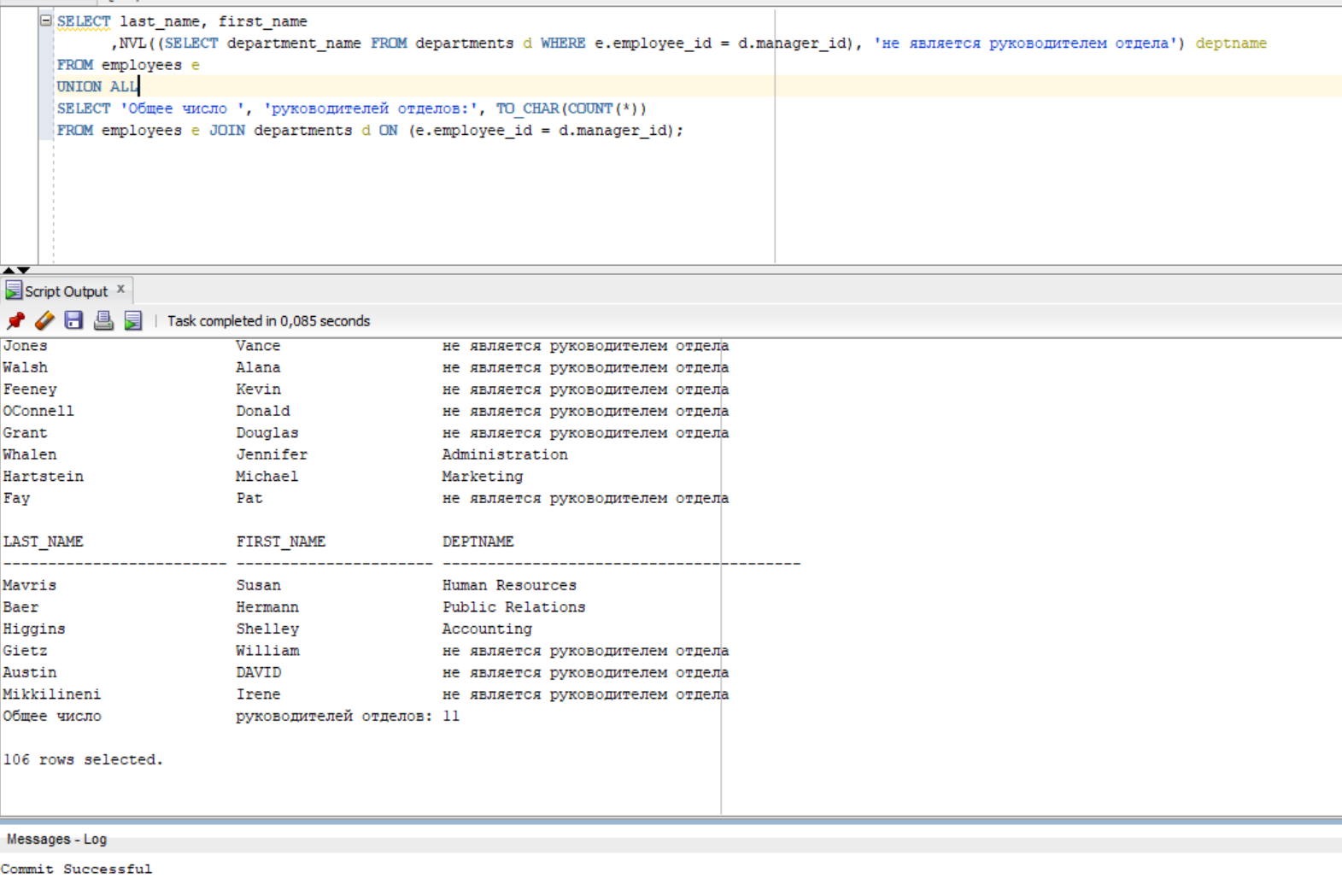


Используя SET-оператор, создайте запрос для определения номеров и идентификаторов должностей сотрудников, чья текущая должность совпадает с когда-либо занимаемой им ранее. Отсортируйте результат по убыванию номера сотрудника.

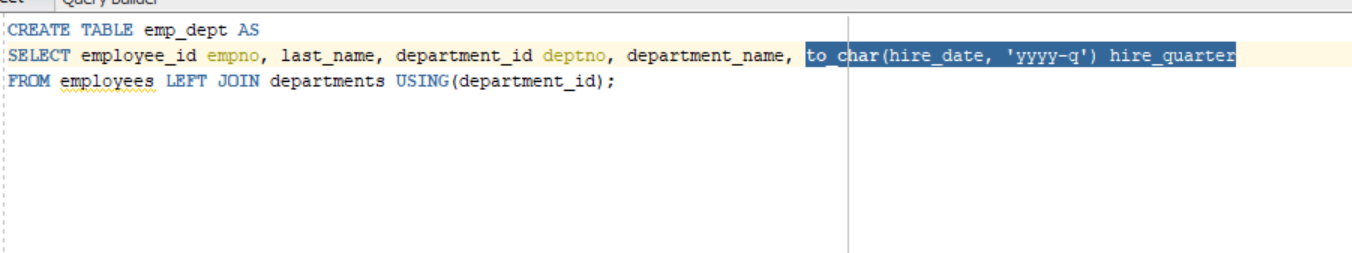




Используя SET-оператор и коррелированный подзапрос в разделе SELECT, создайте запрос для вывода 5 фамилий и имён всех сотрудников. Если сотрудник является руководителем какого-либо отдела, в третьем столбце для него необходимо вывести название этого отдела, если нет – строку «не является руководителем отдела». В последней строке результата выведите общее количество сотрудников, являющихся руководителями отделов.



Создайте таблицу EMP\_DEPT на основе содержимого таблиц EMPLOYEES и DEPARTMENTS. Таблица должна содержать 5 столбцов: номер сотрудника, фамилия, номер отдела, к которому он приписан, наименование отдела, квартал и год, в котором сотрудник был принят на работу. Столбцы с номерами сотрудника и отдела должны иметь названия empno и deptno соответственно, последний столбец - hire\_quarter



# Конспект

Оператор SELECT — один из ключевых элементов языка манипулирования данными (DML) в SQL. Он используется для выборки данных из одной или нескольких таблиц. SELECT позволяет указывать, какие поля выбрать, фильтровать строки с помощью условия WHERE, группировать их с помощью GROUP BY и HAVING, сортировать с помощью ORDER BY, объединять выборки с помощью UNION и ограничивать количество строк с помощью OFFSET и FETCH.

Ключевое слово DISTINCT позволяет исключить повторяющиеся строки из результатов. По умолчанию используется ALL — вывод всех строк, даже если они дублируются.

FROM указывает источник данных: это может быть таблица, представление, подзапрос, производная таблица, общее табличное выражение или хранимая процедура. Для удобства и читаемости можно использовать псевдонимы источников.

WHERE фильтрует строки, выбирая только те, что соответствуют заданному условию. GROUP BY используется для группировки данных по определённым полям, а HAVING позволяет задать фильтрацию уже сгруппированных данных. ORDER BY задаёт порядок следования строк в результате. OFFSET позволяет пропустить заданное количество строк, а FETCH — ограничить число возвращаемых строк, что удобно для постраничной выборки

Производные таблицы — это вложенные запросы внутри FROM, которые можно использовать как обычные таблицы. Они полезны, если нужно упростить запрос или избежать повторных вычислений, как, например, при решении квадратных уравнений, где удобно предварительно вычислить дискриминант и знаменатель в подзапросе, а затем использовать их в основном запросе. В производных таблицах обязательно давать имена всем столбцам, особенно если они получаются в результате выражений. Таким образом, SELECT — мощный инструмент SQL, обеспечивающий гибкое извлечение и обработку данных.

15:50

Соединения (JOIN) в SQL позволяют объединять данные из двух и более источников — обычно таблиц, которые называют левыми и правыми. Соединения выполняются по определённому условию (join condition), чтобы определить, какие строки из таблиц должны быть объединены в результирующий набор данных. Один результат соединения можно использовать как источник для следующего соединения, и так далее. Существует несколько типов соединений, таких как INNER JOIN (внутреннее), OUTER JOIN (внешние: LEFT, RIGHT, FULL), NATURAL JOIN, а также CROSS JOIN (перекрёстное), каждое из которых имеет особенности синтаксиса и правила выполнения.

Допустим, у нас есть две таблицы: PROGRAMMER с колонками LANG\_ID и PNAME, и LANG с колонками LANG\_ID и LNAME. Внутреннее соединение (INNER JOIN) выбирает только те строки, которые удовлетворяют условию соединения — например, LANG\_ID в обеих таблицах совпадает. При запросе SELECT \* FROM PROGRAMMER P INNER JOIN LANG L ON P.LANG\_ID = L.LANG\_ID будут выбраны строки, где LANG\_ID совпадает, и результат будет включать значения из обеих таблиц. Ключевое слово INNER можно опустить, результат будет тем же.

Левое внешнее соединение (LEFT OUTER JOIN) включает в результат все строки из левой таблицы и только те строки правой таблицы, для которых найдено соответствие по условию соединения. Если соответствия нет, поля правой таблицы будут содержать NULL. Например, SELECT \* FROM PROGRAMMER P LEFT OUTER JOIN LANG L ON P.LANG\_ID = L.LANG\_ID покажет всех программистов, даже тех, чьи LANG\_ID не найдены в таблице LANG.

Правое внешнее соединение (RIGHT OUTER JOIN) работает аналогично, но включает все строки из правой таблицы и только соответствующие строки из левой. Например, запрос SELECT \* FROM PROGRAMMER P RIGHT OUTER JOIN LANG L ON P.LANG\_ID = L.LANG\_ID покажет все языки, включая те, которыми никто из программистов не пользуется.

Полное внешнее соединение (FULL OUTER JOIN) возвращает объединение левого и правого внешних соединений, т.е. все строки из обеих таблиц, объединяя совпадающие и добавляя NULL в местах, где соответствий нет. SELECT \* FROM PROGRAMMER P FULL OUTER JOIN LANG L ON P.LANG\_ID = L.LANG\_ID покажет всех программистов и все языки, независимо от наличия соответствия.

Соединения можно делать с помощью явного условия ON, указывая, по каким полям происходит сравнение. Чаще всего используется эквисоединение — сравнение на равенство. Например, SELECT \* FROM customers c JOIN sales s ON s.cust\_id = [c.id](https://c.id/) выбирает всех клиентов, сделавших покупку. LEFT JOIN может быть использован, чтобы показать также тех, кто покупок не совершал.

Можно использовать соединение именованных столбцов с помощью USING, когда имена столбцов одинаковые в обеих таблицах. Например, SELECT \* FROM PROGRAMMER P INNER JOIN LANG L USING (LANG\_ID) — аналог предыдущего INNER JOIN, но короче и не дублирует столбец LANG\_ID в результате.

NATURAL JOIN — это ещё более автоматизированный способ соединения, при котором система сама соединяет таблицы по всем одноимённым столбцам. Например, SELECT \* FROM PROGRAMMER P NATURAL JOIN LANG L эквивалентен INNER JOIN с USING (LANG\_ID). Однако NATURAL JOIN стоит использовать с осторожностью — если структуры таблиц изменятся, результат может быть неожиданным.

Неявные соединения использовались раньше и задаются через запятую в FROM и условие в WHERE. Например, SELECT \* FROM PROGRAMMER P, LANG L WHERE P.LANG\_ID = LANG.LANG\_ID — эквивалент INNER JOIN, но менее нагляден и не рекомендуется к использованию.

Перекрёстные соединения (CROSS JOIN) создают декартово произведение двух таблиц — каждая строка одной соединяется с каждой строкой другой. Они полезны, когда нужно получить все возможные комбинации значений. Например, SELECT \* FROM PROGRAMMER CROSS JOIN LANG покажет список всех возможных пар программистов и языков. Также можно использовать их для генерации всех комбинаций атрибутов товара — например, материал, размер и цвет: SELECT [m.name](https://m.name/), s.size, [c.name](https://c.name/) FROM materials m CROSS JOIN sizes s CROSS JOIN colors c. Это создаст набор всех возможных сочетаний трёх параметров.

Оператор WHERE в SQL используется для фильтрации строк, которые должны быть выбраны, обновлены или удалены в таблице. Он применяется после ключевого слова FROM и указывает условие, которому должны соответствовать строки. Основной синтаксис выглядит так: SELECT столбцы FROM таблица WHERE условие;. Условием может быть сравнение значений, например WHERE age = 25, WHERE salary > 50000, WHERE status <> 'inactive' и так далее. Также можно использовать логические операторы AND, OR и NOT для объединения нескольких условий, например WHERE age > 18 AND status = 'active'.

Для проверки значений в диапазоне применяется BETWEEN, например WHERE age BETWEEN 18 AND 25. Если нужно проверить наличие значения в списке, используется IN, например WHERE department IN ('HR', 'IT', 'Sales'), а для исключения — NOT IN. С помощью LIKE можно искать строки по шаблону, например WHERE name LIKE 'A%', где % обозначает любое количество любых символов, а \_ — ровно один символ. Проверку на NULL значения выполняют с помощью IS NULL и IS NOT NULL, например WHERE email IS NOT NULL.

Оператор WHERE используется не только в запросах SELECT, но и в UPDATE и DELETE. Например, UPDATE employees SET status = 'inactive' WHERE last\_login < '2023-01-01'; обновит только тех сотрудников, которые не заходили с начала 2023 года. А команда DELETE FROM users WHERE status = 'banned'; удалит всех пользователей со статусом "banned". Без WHERE эти команды повлияют на все строки таблицы, поэтому его отсутствие может привести к потере данных. Если нужно фильтровать группы данных, а не отдельные строки, используют HAVING, так как WHERE не работает с агрегатными функциями (SUM, COUNT, AVG и т.п.).

Таким образом, WHERE — ключевой инструмент для выборки и изменения только нужных данных в SQL-запросах.

В этой главе рассматриваются встроенные скалярные функции SQL, используемые для фильтрации записей, выбора данных, построения индексов и других целей. Контекстные переменные позволяют сохранять и получать различные значения, как системные, так и пользовательские. Они принадлежат к пространствам имен: SYSTEM (только для чтения), USER\_SESSION (существуют в течение подключения), USER\_TRANSACTION (существуют в течение транзакции), DDL\_TRIGGER (используются в триггерах), AUTHDATA (содержит данные об аутентификации). Функция RDB$GET\_CONTEXT возвращает значение контекстной переменной, а RDB$SET\_CONTEXT устанавливает или удаляет переменную. Всего можно создать до 1000 переменных на соединение. Пространства имен и имена переменных чувствительны к регистру.

Среди системных переменных в пространстве имен SYSTEM можно выделить: FULL\_VERSION (версия СУБД), EDITION (редакция), DB\_NAME (путь к базе), CLIENT\_ADDRESS и CLIENT\_PROCESS (информация о клиенте), CURRENT\_USER и CURRENT\_ROLE (глобальные переменные), SESSION\_TIMEZONE (текущий часовой пояс). Обращение к несуществующим переменным в SYSTEM вызывает ошибку, а в пользовательских пространствах — возвращает NULL.

Математические функции: ABS(x) — модуль, SIN(x), COS(x), TAN(x), COT(x) — тригонометрические функции, ACOS(x), ASIN(x), ATAN(x) — обратные тригонометрические, CEIL(x) и FLOOR(x) — округление вверх/вниз, LN(x), LOG(x, b), LOG10(x) — логарифмы, MOD(x, y) — остаток от деления, PI() — число Пи, POWER(x, y) — возведение в степень, RAND() — случайное число, ROUND(x, p) — округление до p знаков, SQRT(x) — квадратный корень. При NULL на входе — возвращается NULL.

Функции работы со строками: ASCII\_CHAR(n) — символ по ASCII-коду, ASCII\_VAL(str) — код первого символа, OCTET\_LENGTH(str) — длина строки в байтах, CHARACTER\_LENGTH(str) или CHAR\_LENGTH — количество символов. LOWER(str) и UPPER(str) — перевод регистра. POSITION(substring IN string) — позиция подстроки. REPLACE(str, old, new) — замена подстроки. SUBSTRING(str FROM pos FOR len) — извлечение подстроки. TRIM([LEADING|TRAILING|BOTH] [chars] FROM str) — удаление символов с начала и/или конца строки.

Функции работы с датами и временем: DATEADD(unit, value, date) — прибавление к дате. DATEDIFF(unit FROM date1 TO date2) — разница между датами. EXTRACT(unit FROM date) — извлечение элемента даты (день, месяц, год и др.). Поддерживаются типы DATE, TIME и TIMESTAMP. При неправильной комбинации типа данных и единицы времени возникает ошибка.

Функция CAST преобразует данные из одного типа в другой: CAST(value AS type). NULL преобразуется в NULL любого типа. Допустимо преобразование между числовыми, строковыми и временными типами, если значение соответствует формату.

Функции побитовых операций: BIN\_AND(x1,x2,...) — логическое И, BIN\_OR(x1,x2,...) — ИЛИ, BIN\_XOR(x1,x2,...) — исключающее ИЛИ, BIN\_NOT(x) — отрицание, BIN\_SHL(x, n) — сдвиг влево, BIN\_SHR(x, n) — сдвиг вправо. Возвращают NULL, если один из аргументов NULL.

Работа с UUID: CHAR\_TO\_UUID(str) — преобразует ASCII UUID в компактную форму. UUID\_TO\_CHAR(uuid) — обратно. GEN\_UUID() — генерирует уникальный UUID.

Условные функции: CASE позволяет выбирать результат по значению или условию. Существует два синтаксиса: обычный CASE (сравнение значений) и поисковый CASE (логические выражения). Если совпадение не найдено, возвращается значение из ELSE или NULL. Функция COALESCE возвращает первое из непустых значений в списке.