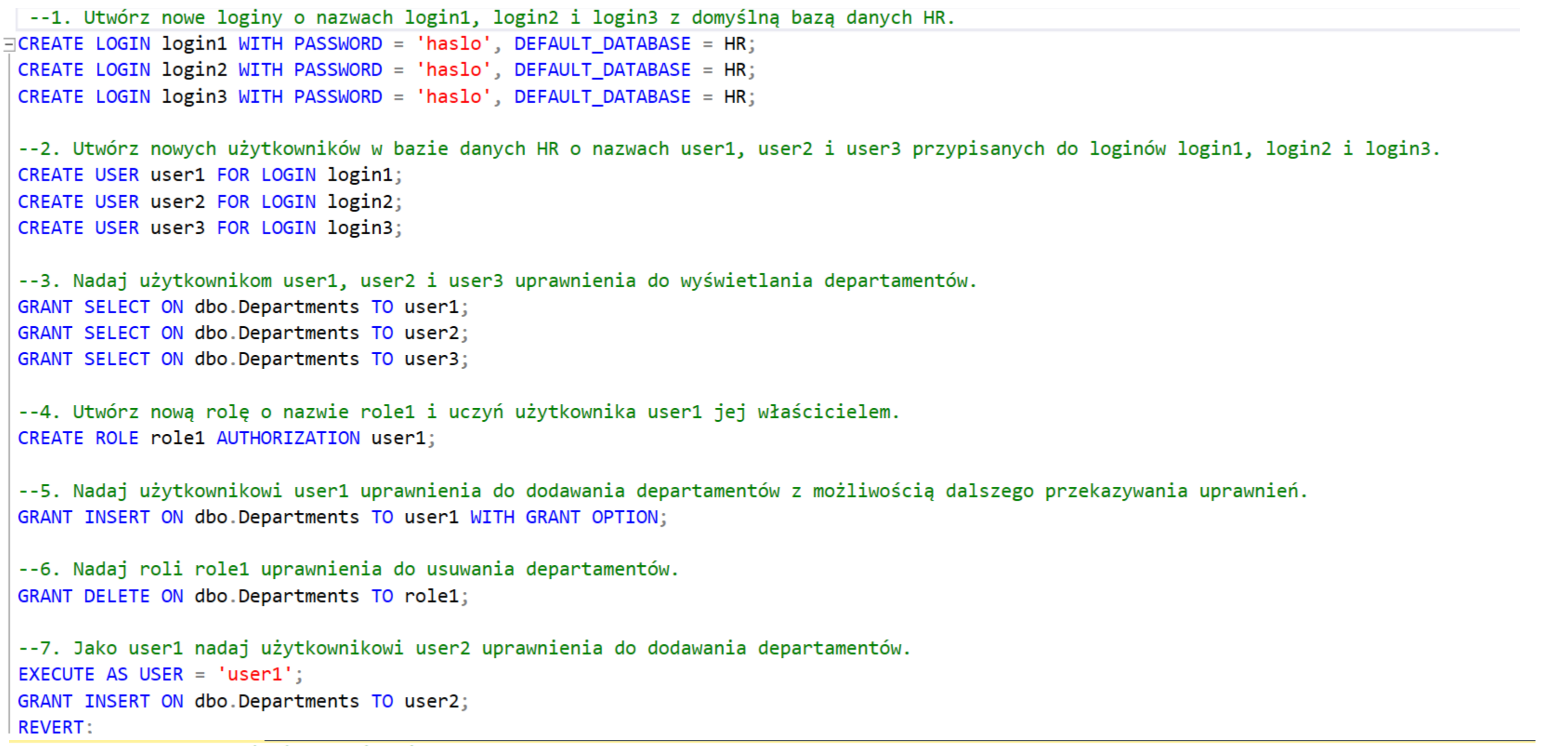
Pavel Radkevich
Bartłomiej Czerwiński
Adam Rosiak
Amadeusz Sitnicki

Środa: 17:00 – 20:15
03.11.2022

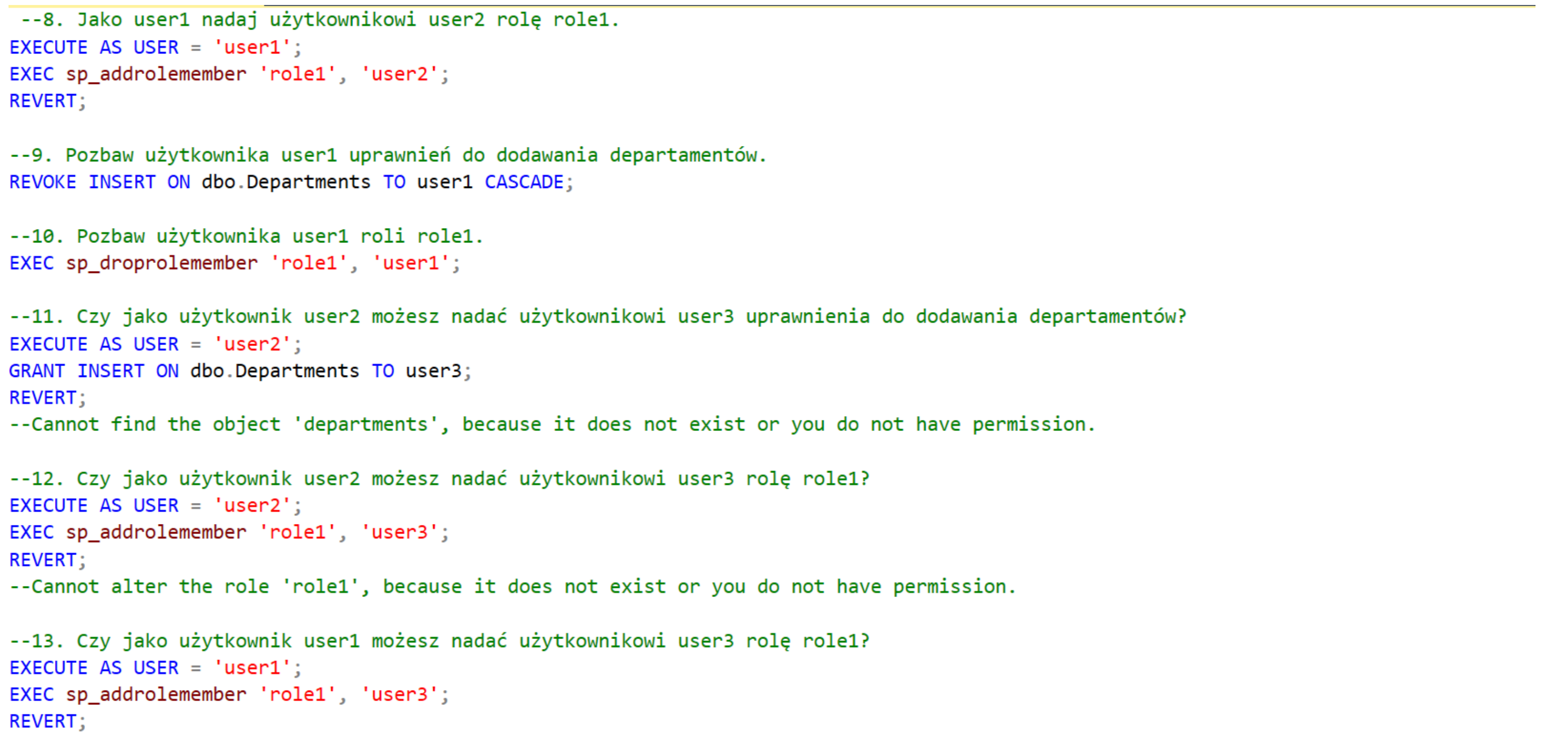
Administrowanie i programowanie baz danych

Zadanie 4

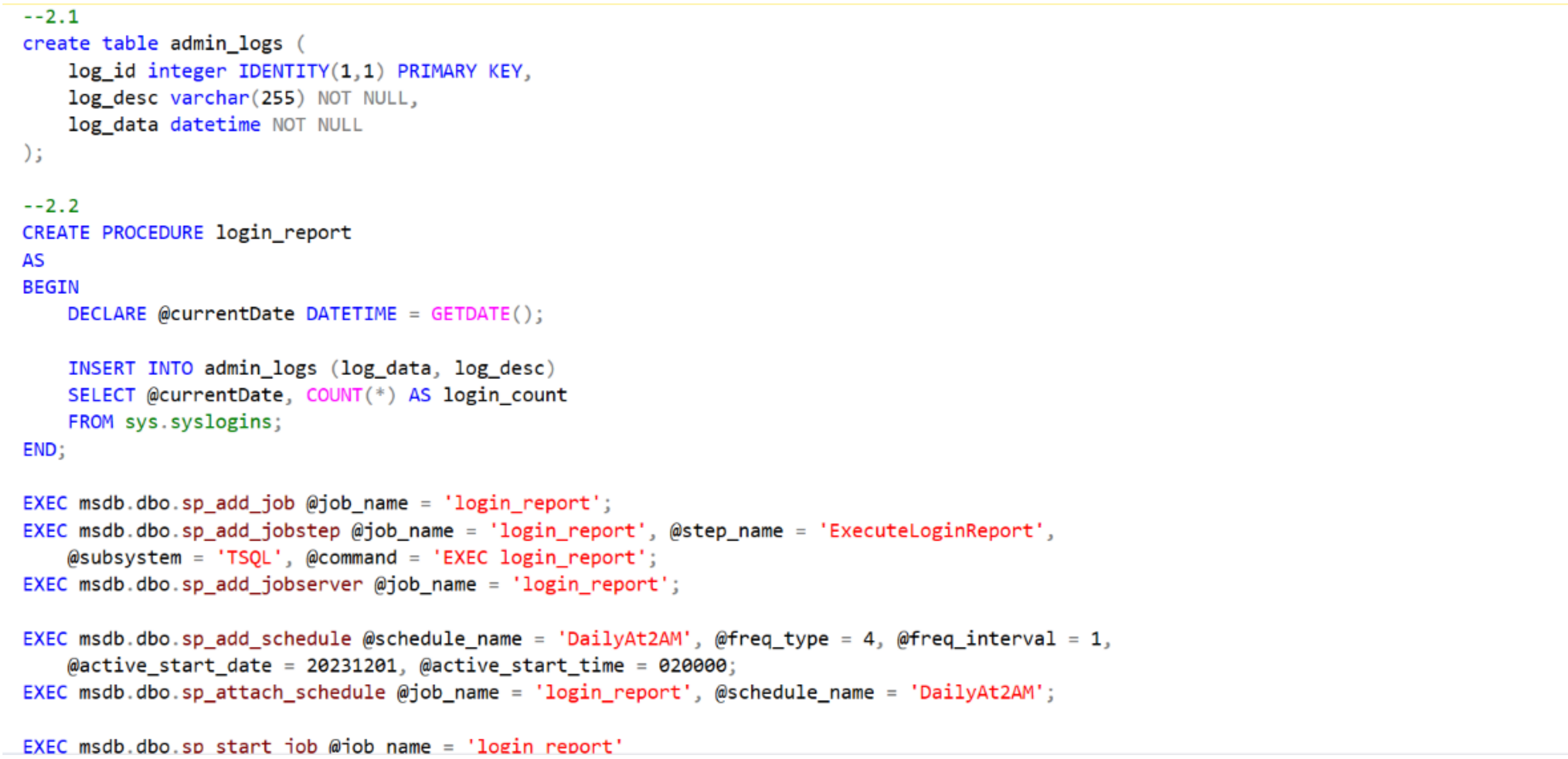
1. Microsoft SQL Server



```sql
--1. Utwórz nowe loginy o nazwach login1, login2 i login3 z domyślną bazą danych HR.
CREATE LOGIN login1 WITH PASSWORD = 'haslo', DEFAULT_DATABASE = HR;
CREATE LOGIN login2 WITH PASSWORD = 'haslo', DEFAULT_DATABASE = HR;
CREATE LOGIN login3 WITH PASSWORD = 'haslo', DEFAULT_DATABASE = HR;

--2. Utwórz nowych użytkowników w bazie danych HR o nazwach user1, user2 i user3 przypisanych do loginów login1, login2 i login3.
CREATE USER user1 FOR LOGIN login1;
CREATE USER user2 FOR LOGIN login2;
CREATE USER user3 FOR LOGIN login3;

--3. Nadaj użytkownikom user1, user2 i user3 uprawnienia do wyświetlania departamentów.
GRANT SELECT ON dbo.Departments TO user1;
GRANT SELECT ON dbo.Departments TO user2;
GRANT SELECT ON dbo.Departments TO user3;

--4. Utwórz nową rolę o nazwie role1 i uczyń użytkownika user1 jej właścicielem.
CREATE ROLE role1 AUTHORIZATION user1;

--5. Nadaj użytkownikowi user1 uprawnienia do dodawania departamentów z możliwością dalszego przekazywania uprawnień.
GRANT INSERT ON dbo.Departments TO user1 WITH GRANT OPTION;

--6. Nadaj roli role1 uprawnienia do usuwania departamentów.
GRANT DELETE ON dbo.Departments TO role1;

--7. Jako user1 nadaj użytkownikowi user2 uprawnienia do dodawania departamentów.
EXECUTE AS USER = 'user1';
GRANT INSERT ON dbo.Departments TO user2;
REVERT;
```

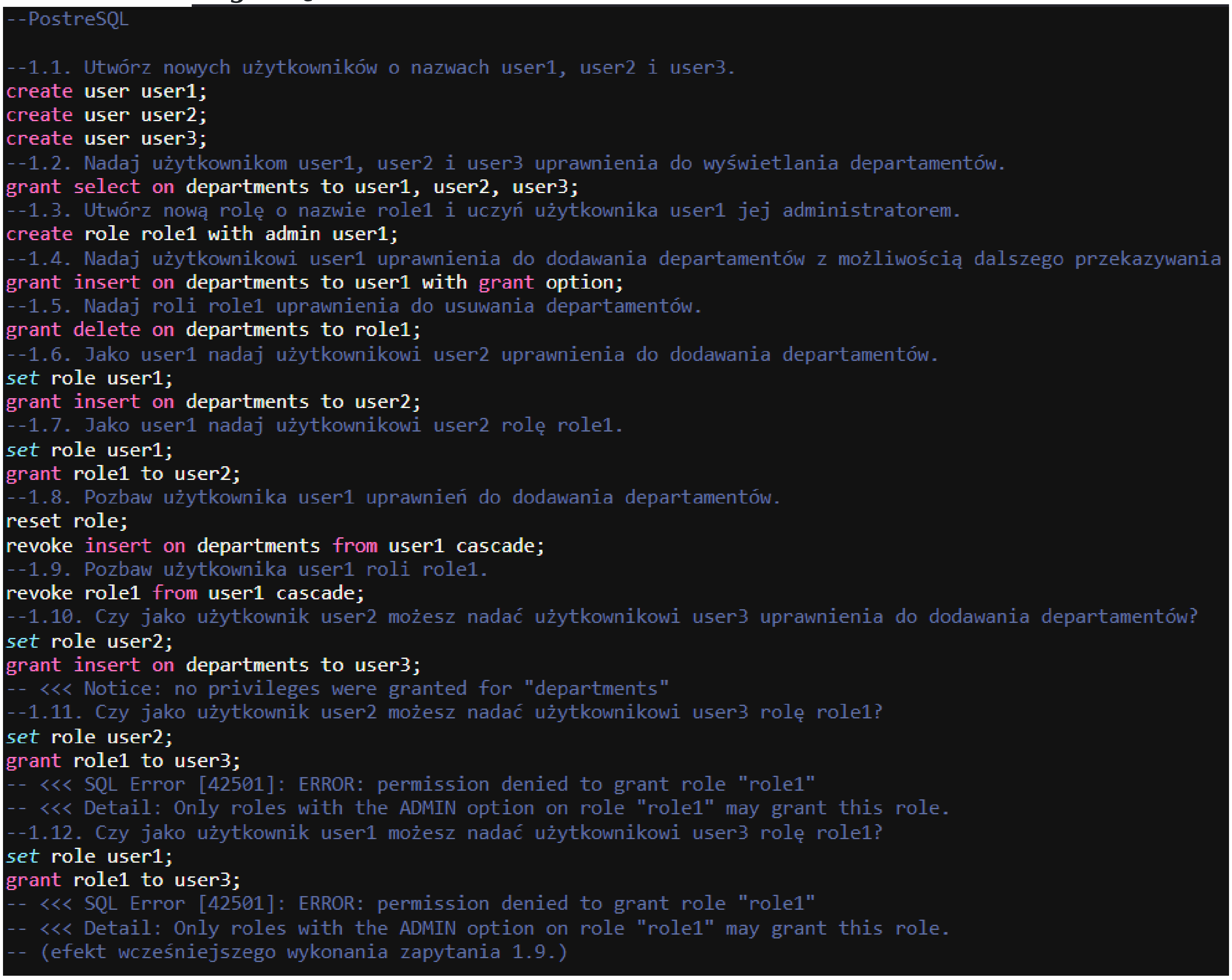


```sql
--8. Jako user1 nadaj użytkownikowi user2 rolę role1.
EXECUTE AS USER = 'user1';
EXEC sp_addrolemember 'role1', 'user2';
REVERT;

--9. Pozbaw użytkownika user1 uprawnień do dodawania departamentów.
REVOKE INSERT ON dbo.Departments TO user1 CASCADE;

--10. Pozbaw użytkownika user1 roli role1.
EXEC sp_droprolemember 'role1', 'user1';

--11. Czy jako użytkownik user2 możesz nadać użytkownikowi user3 uprawnienia do dodawania departamentów?
EXECUTE AS USER = 'user2';
GRANT INSERT ON dbo.Departments TO user3;
REVERT;
--Cannot find the object 'departments', because it does not exist or you do not have permission.

--12. Czy jako użytkownik user2 możesz nadać użytkownikowi user3 rolę role1?
EXECUTE AS USER = 'user2';
EXEC sp_addrolemember 'role1', 'user3';
REVERT;
--Cannot alter the role 'role1', because it does not exist or you do not have permission.

--13. Czy jako użytkownik user1 możesz nadać użytkownikowi user3 rolę role1?
EXECUTE AS USER = 'user1';
EXEC sp_addrolemember 'role1', 'user3';
REVERT;
```

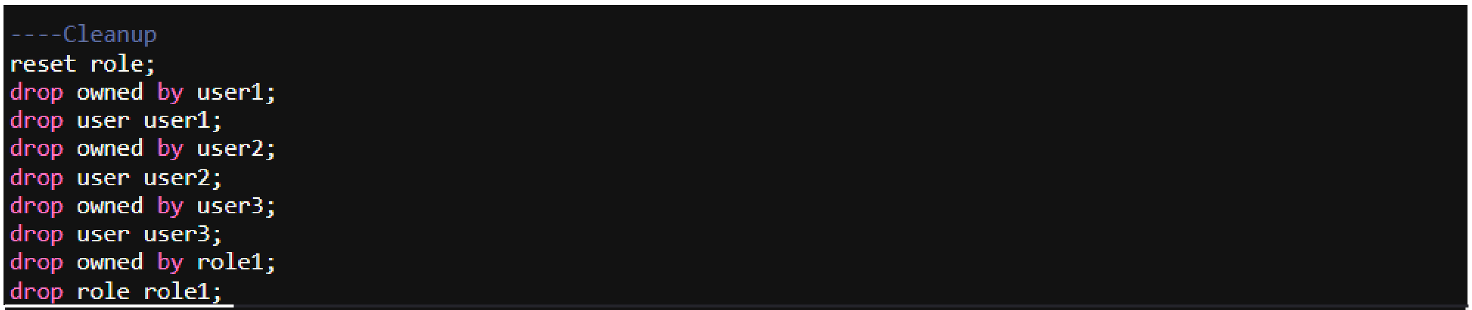


```sql
--2.1
create table admin_logs (
    log_id integer IDENTITY(1,1) PRIMARY KEY,
    log_desc varchar(255) NOT NULL,
    log_data datetime NOT NULL
);

--2.2
CREATE PROCEDURE login_report
AS
BEGIN
    DECLARE @currentDate DATETIME = GETDATE();

    INSERT INTO admin_logs (log_data, log_desc)
    SELECT @currentDate, COUNT(*) AS login_count
    FROM sys.syslogins;
END;

EXEC msdb.dbo.sp_add_job @job_name = 'login_report';
EXEC msdb.dbo.sp_add_jobstep @job_name = 'login_report', @step_name = 'ExecuteLoginReport',
    @subsystem = 'TSQL', @command = 'EXEC login_report';
EXEC msdb.dbo.sp_add_jobserver @job_name = 'login_report';

EXEC msdb.dbo.sp_add_schedule @schedule_name = 'DailyAt2AM', @freq_type = 4, @freq_interval = 1,
    @active_start_date = 20231201, @active_start_time = 020000;
EXEC msdb.dbo.sp_attach_schedule @job_name = 'login_report', @schedule_name = 'DailyAt2AM';

EXEC msdb.dbo.sp_start_job @job_name = 'login_report'
```

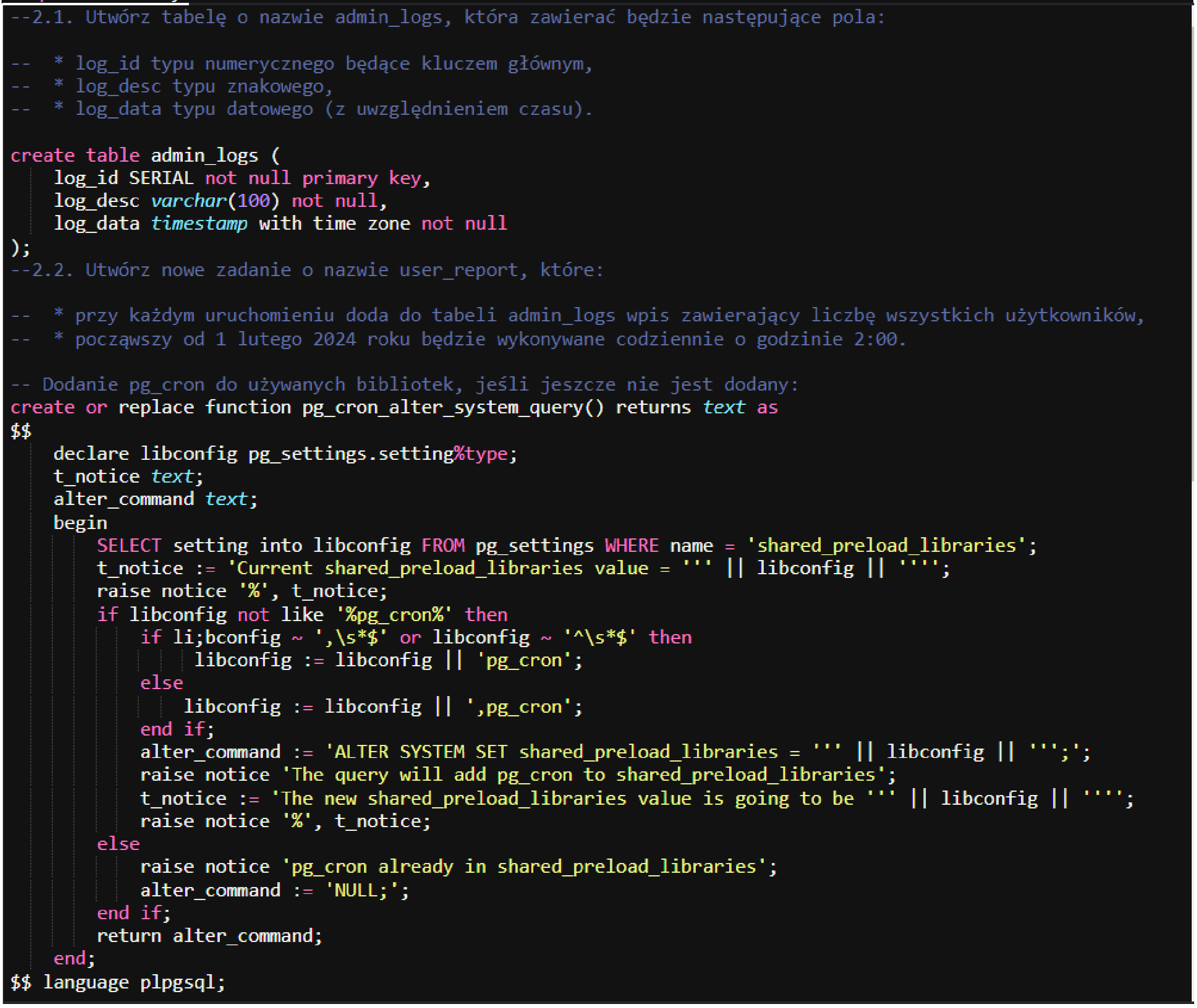
2. PostgreSQL



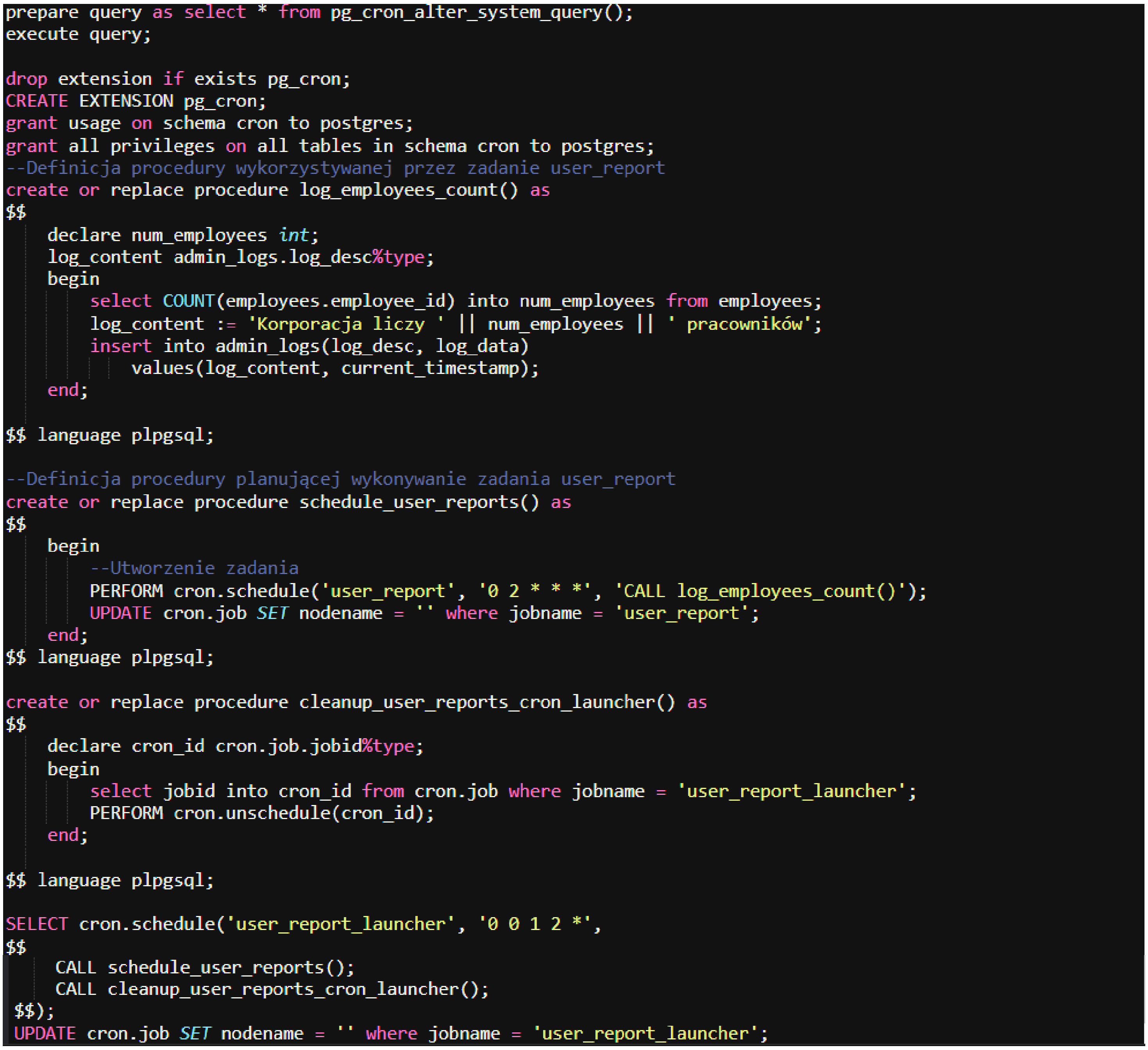
```sql
--PostreSQL

--1.1. Utwórz nowych użytkowników o nazwach user1, user2 i user3.
create user user1;
create user user2;
create user user3;
--1.2. Nadaj użytkownikom user1, user2 i user3 uprawnienia do wyświetlania departamentów.
grant select on departments to user1, user2, user3;
--1.3. Utwórz nową rolę o nazwie role1 i uczyń użytkownika user1 jej administratorem.
create role role1 with admin user1;
--1.4. Nadaj użytkownikowi user1 uprawnienia do dodawania departamentów z możliwością dalszego przekazywania
grant insert on departments to user1 with grant option;
--1.5. Nadaj roli role1 uprawnienia do usuwania departamentów.
grant delete on departments to role1;
--1.6. Jako user1 nadaj użytkownikowi user2 uprawnienia do dodawania departamentów.
set role user1;
grant insert on departments to user2;
--1.7. Jako user1 nadaj użytkownikowi user2 rolę role1.
set role user1;
grant role1 to user2;
--1.8. Pozbaw użytkownika user1 uprawnień do dodawania departamentów.
reset role;
revoke insert on departments from user1 cascade;
--1.9. Pozbaw użytkownika user1 roli role1.
revoke role1 from user1 cascade;
--1.10. Czy jako użytkownik user2 możesz nadać użytkownikowi user3 uprawnienia do dodawania departamentów?
set role user2;
grant insert on departments to user3;
-- <<< Notice: no privileges were granted for "departments"
--1.11. Czy jako użytkownik user2 możesz nadać użytkownikowi user3 rolę role1?
set role user2;
grant role1 to user3;
-- <<< SQL Error [42501]: ERROR: permission denied to grant role "role1"
-- <<< Detail: Only roles with the ADMIN option on role "role1" may grant this role.
--1.12. Czy jako użytkownik user1 możesz nadać użytkownikowi user3 rolę role1?
set role user1;
grant role1 to user3;
-- <<< SQL Error [42501]: ERROR: permission denied to grant role "role1"
-- <<< Detail: Only roles with the ADMIN option on role "role1" may grant this role.
-- (efekt wcześniejszego wykonania zapytania 1.9.)
```



```sql
----Cleanup
reset role;
drop owned by user1;
drop user user1;
drop owned by user2;
drop user user2;
drop owned by user3;
drop user user3;
drop owned by role1;
drop role role1;
```

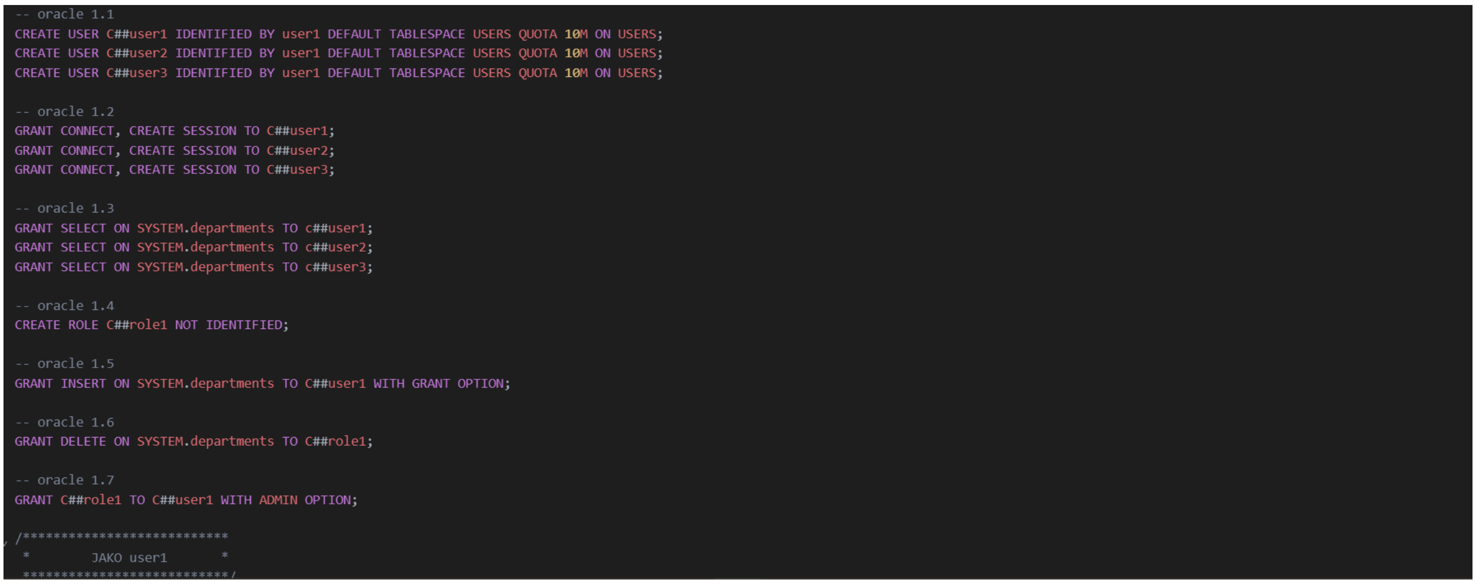


```sql
--2.1. Utwórz tabelę o nazwie admin_logs, która zawierać będzie następujące pola:

--  * log_id typu numerycznego będące kluczem głównym,
--  * log_desc typu znakowego,
--  * log_data typu datowego (z uwzględnieniem czasu).

create table admin_logs (
    log_id SERIAL not null primary key,
    log_desc varchar(100) not null,
    log_data timestamp with time zone not null
);
--2.2. Utwórz nowe zadanie o nazwie user_report, które:

--  * przy każdym uruchomieniu doda do tabeli admin_logs wpis zawierający liczbę wszystkich użytkowników,
--  * począwszy od 1 lutego 2024 roku będzie wykonywane codziennie o godzinie 2:00.

-- Dodanie pg_cron do używanych bibliotek, jeśli jeszcze nie jest dodany:
create or replace function pg_cron_alter_system_query() returns text as
$$
    declare libconfig pg_settings.setting%type;
    t_notice text;
    alter_command text;
    begin
        SELECT setting into libconfig FROM pg_settings WHERE name = 'shared_preload_libraries';
        t_notice := 'Current shared_preload_libraries value = ''' || libconfig || '''';
        raise notice '%', t_notice;
        if libconfig not like '%pg_cron%' then
            if li;bconfig ~ ',\s*$' or libconfig ~ '^\s*$' then
                 libconfig := libconfig || 'pg_cron';
            else
                libconfig := libconfig || ',pg_cron';
            end if;
            alter_command := 'ALTER SYSTEM SET shared_preload_libraries = ''' || libconfig || ''';';
            raise notice 'The query will add pg_cron to shared_preload_libraries';
            t_notice := 'The new shared_preload_libraries value is going to be ''' || libconfig || '''';
            raise notice '%', t_notice;
        else
            raise notice 'pg_cron already in shared_preload_libraries';
            alter_command := 'NULL;';
        end if;
        return alter_command;
    end;
$$ language plpgsql;
```

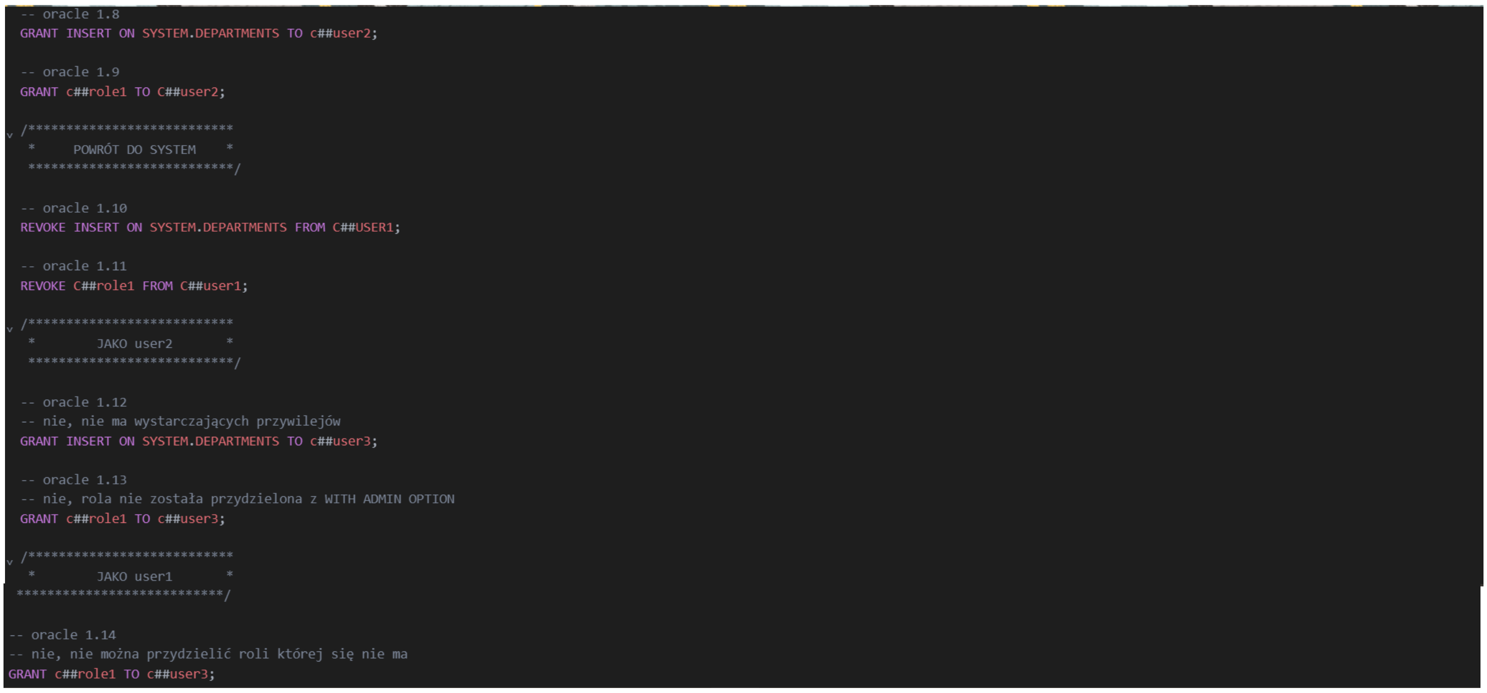


```sql
prepare query as select * from pg_cron_alter_system_query();
execute query;

drop extension if exists pg_cron;
CREATE EXTENSION pg_cron;
grant usage on schema cron to postgres;
grant all privileges on all tables in schema cron to postgres;
--Definicja procedury wykorzystywanej przez zadanie user_report
create or replace procedure log_employees_count() as
$$
    declare num_employees int;
    log_content admin_logs.log_desc%type;
    begin
        select COUNT(employees.employee_id) into num_employees from employees;
        log_content := 'Korporacja liczy ' || num_employees || ' pracowników';
        insert into admin_logs(log_desc, log_data)
            values(log_content, current_timestamp);
    end;

$$ language plpgsql;

--Definicja procedury planującej wykonywanie zadania user_report
create or replace procedure schedule_user_reports() as
$$
    begin
        --Utworzenie zadania
        PERFORM cron.schedule('user_report', '0 2 * * *', 'CALL log_employees_count()');
        UPDATE cron.job SET nodename = '' where jobname = 'user_report';
    end;
$$ language plpgsql;

create or replace procedure cleanup_user_reports_cron_launcher() as
$$
    declare cron_id cron.job.jobid%type;
    begin
        select jobid into cron_id from cron.job where jobname = 'user_report_launcher';
        PERFORM cron.unschedule(cron_id);
    end;

$$ language plpgsql;

SELECT cron.schedule('user_report_launcher', '0 0 1 2 *',
$$
     CALL schedule_user_reports();
     CALL cleanup_user_reports_cron_launcher();
 $$);
 UPDATE cron.job SET nodename = '' where jobname = 'user_report_launcher';
```

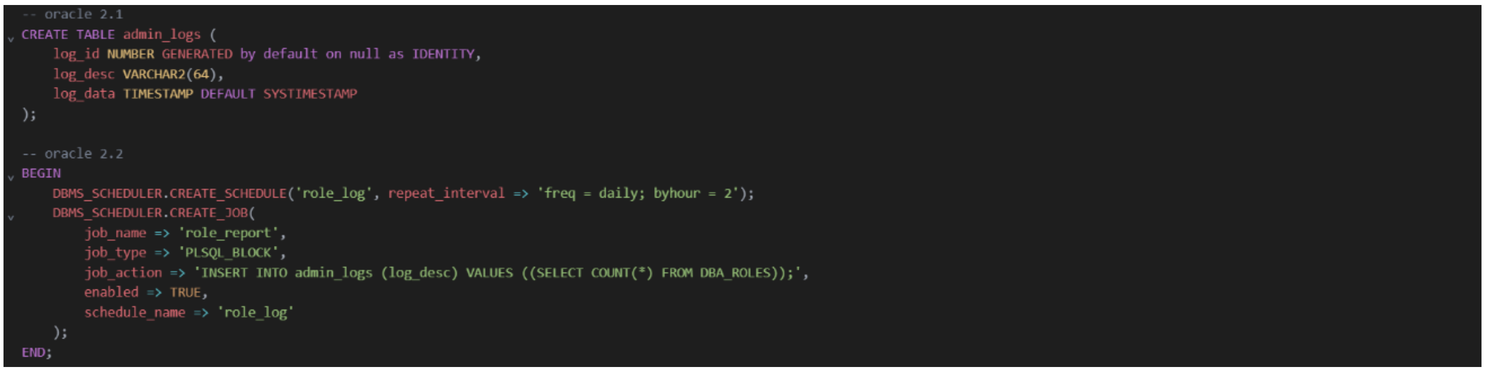
3. Oracle Database



```sql
-- oracle 1.1
CREATE USER C##user1 IDENTIFIED BY user1 DEFAULT TABLESPACE USERS QUOTA 10M ON USERS;
CREATE USER C##user2 IDENTIFIED BY user1 DEFAULT TABLESPACE USERS QUOTA 10M ON USERS;
CREATE USER C##user3 IDENTIFIED BY user1 DEFAULT TABLESPACE USERS QUOTA 10M ON USERS;

-- oracle 1.2
GRANT CONNECT, CREATE SESSION TO C##user1;
GRANT CONNECT, CREATE SESSION TO C##user2;
GRANT CONNECT, CREATE SESSION TO C##user3;

-- oracle 1.3
GRANT SELECT ON SYSTEM.departments TO c##user1;
GRANT SELECT ON SYSTEM.departments TO c##user2;
GRANT SELECT ON SYSTEM.departments TO c##user3;

-- oracle 1.4
CREATE ROLE C##role1 NOT IDENTIFIED;

-- oracle 1.5
GRANT INSERT ON SYSTEM.departments TO C##user1 WITH GRANT OPTION;

-- oracle 1.6
GRANT DELETE ON SYSTEM.departments TO C##role1;

-- oracle 1.7
GRANT C##role1 TO C##user1 WITH ADMIN OPTION;

/**************************
 *        JAKO user1      *
 **************************/
```



```sql
-- oracle 1.8
GRANT INSERT ON SYSTEM.DEPARTMENTS TO c##user2;

-- oracle 1.9
GRANT c##role1 TO C##user2;

/***************************
 *     POWRÓT DO SYSTEM    *
 ***************************/

-- oracle 1.10
REVOKE INSERT ON SYSTEM.DEPARTMENTS FROM C##USER1;

-- oracle 1.11
REVOKE C##role1 FROM C##user1;

/***************************
 *        JAKO user2       *
 ***************************/

-- oracle 1.12
-- nie, nie ma wystarczających przywilejów
GRANT INSERT ON SYSTEM.DEPARTMENTS TO c##user3;

-- oracle 1.13
-- nie, rola nie została przydzielona z WITH ADMIN OPTION
GRANT c##role1 TO c##user3;

/***************************
 *        JAKO user1       *
***************************/

-- oracle 1.14
-- nie, nie można przydzielić roli której się nie ma
GRANT c##role1 TO c##user3;
```



```sql
-- oracle 2.1
CREATE TABLE admin_logs (
    log_id NUMBER GENERATED by default on null as IDENTITY,
    log_desc VARCHAR2(64),
    log_data TIMESTAMP DEFAULT SYSTIMESTAMP
);

-- oracle 2.2
BEGIN
    DBMS_SCHEDULER.CREATE_SCHEDULE('role_log', repeat_interval => 'freq = daily; byhour = 2');
    DBMS_SCHEDULER.CREATE_JOB(
        job_name => 'role_report',
        job_type => 'PLSQL_BLOCK',
        job_action => 'INSERT INTO admin_logs (log_desc) VALUES ((SELECT COUNT(*) FROM DBA_ROLES));',
        enabled => TRUE,
        schedule_name => 'role_log'
    );
END;
```