|  |
| --- |
| **Хранимые процедуры** |

**Цель работы**

По аналогии с примерами, приведенными в п. 1 , создать в БД ХП, реализующие:

— вставку с пополнением справочников (вставляется информация о студенте, если указанный номер группы отсутствует в БД, запись добавляется в таблицу с перечнем групп) (получаем ссылку на внешний ключ по значению данного из родительской таблицы);

— удаление с очисткой справочников – удаление всех зависимых данных(удаляется информация о студенте, если в его группе нет больше студентов, запись удаляется из таблицы с перечнем групп);

— каскадное удаление (перед удалением записи о группе удаляются записи обо всех студентах этой группы);

— вычисление и возврат значения агрегатной функции (т.к. агрегатная функция дает единственный результат);

— формирование статистики во временной таблице (например, для рассматриваемой БД — для каждого факультета: количество групп, количество обучающихся студентов, количество изучаемых дисциплин, количество сданных дисциплин, средний балл по факультету).

**Вариант 12**

вакансии: название вакансии, организация работодатель, адрес работодателя, диапазон

зарплаты, требования к образованию, Обязанности, график работы, требования обязательные,

желательные, дата выставления вакансии

а. вакансии, имеющие в названии «разработчик», но не начинающиеся на него

б. обязанность, не присутствующая ни в одной вакансии

в. работодатели в Москве, выставившие вакансии и программиста Java и системного

администратора

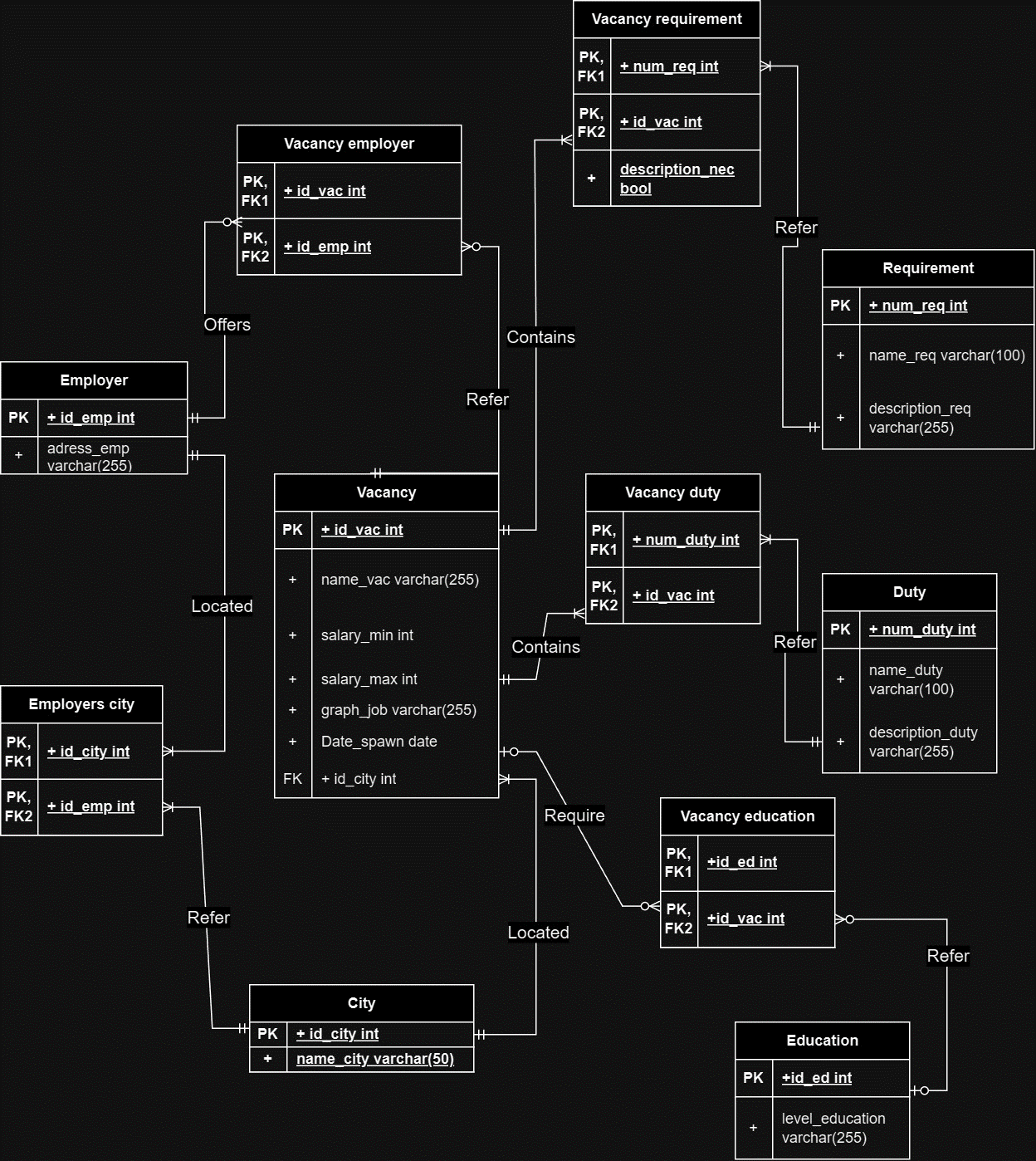
г. вакансия с зарплатой ниже среднего

д. вакансии с максимальным количеством обязательных требований

е. требование, присутствующее во всех вакансиях на переводчика (любые вакансии со словом перевод)

ж. вакансии, которые есть в Санкт-Петербурге или Пскове, но которых нет в Москве

**Физическая модель БД**



1. **Процедура вставки с пополнением справочника.**

DELIMITER //

CREATE PROCEDURE ins\_vacancy (IN name\_city\_ VARCHAR(50),

IN name\_vac\_ VARCHAR(255),

IN salary\_min\_ INT,

IN salary\_max\_ INT,

IN graph\_job\_ VARCHAR(255),

IN date\_spawn\_ DATE)

BEGIN

DECLARE id\_vac\_new INT;

DECLARE id\_city\_new INT;

if exists(select \* from City where name\_city = name\_city\_)

then select id\_city into id\_city\_new from City where name\_city = name\_city\_;

else begin

INSERT INTO City(name\_city) VALUES (name\_city\_);

set id\_city\_new = (select LAST\_INSERT\_ID());

end;

end if;

insert into Vacancy (name\_vac, salary\_min, salary\_max, graph\_job, date\_spawn, id\_city) VALUES (name\_vac\_, salary\_min\_, salary\_max\_, graph\_job\_, date\_spawn\_, id\_city\_new);

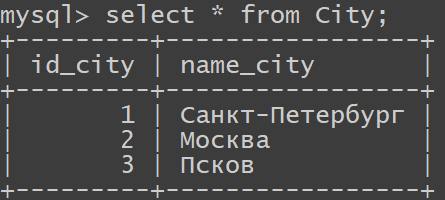
END;//

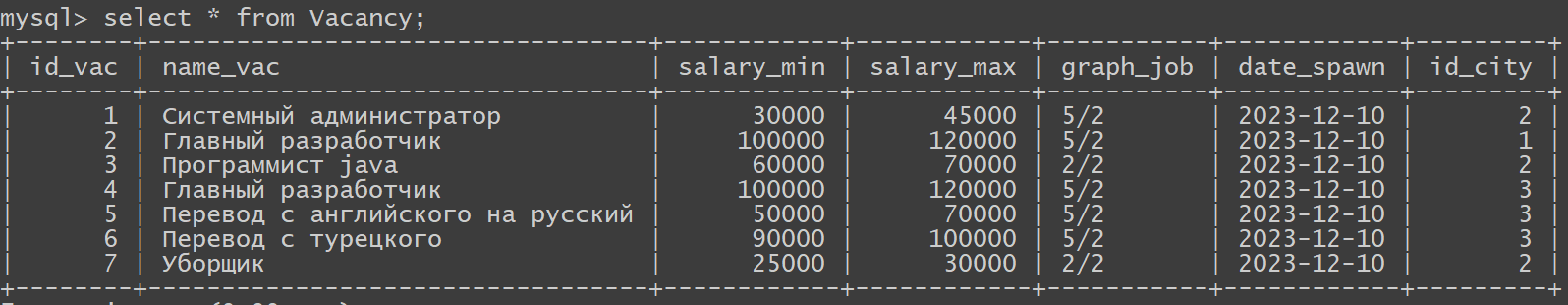
DELIMITER ;

CALL ins\_vacancy('Челябинск', 'Директор', 60500, 150000, '5/2', NOW());

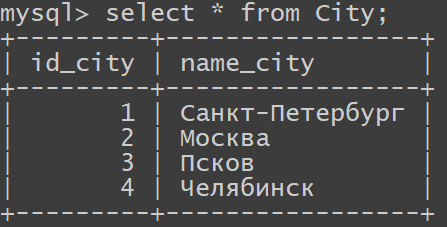
CALL ins\_vacancy('Воркута', 'Директор', 60500, 150000, '5/2', NOW());

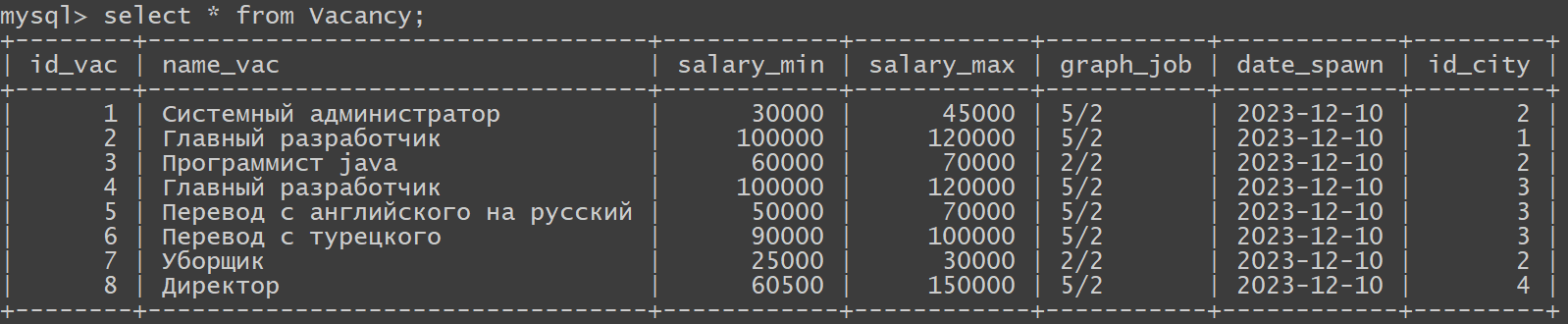
Наборы данных до вызова:





Наборы данных после вызова:





1. **Процедура удаления с очисткой справочника (Удаляется город, если в этом городе нет вакансий и работодателей)**

DELIMITER //

CREATE PROCEDURE del\_vac\_clear\_city (IN id\_vac\_del INT)

BEGIN

DECLARE id\_city\_del INT;

SELECT id\_city INTO id\_city\_del FROM Vacancy where id\_vac = id\_vac\_del;

Delete from Vacancy where id\_vac = id\_vac\_del;

If not exists(select \* from Vacancy WHERE id\_city = id\_city\_del) AND

not exists(select \* from Employer\_city WHERE id\_city = id\_city\_del)

then DELETE from City WHERE id\_city = id\_city\_del;

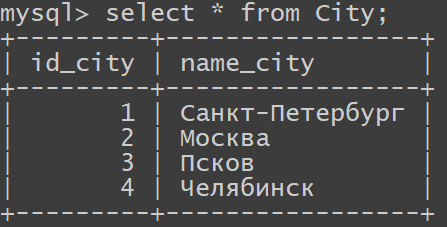
end if;

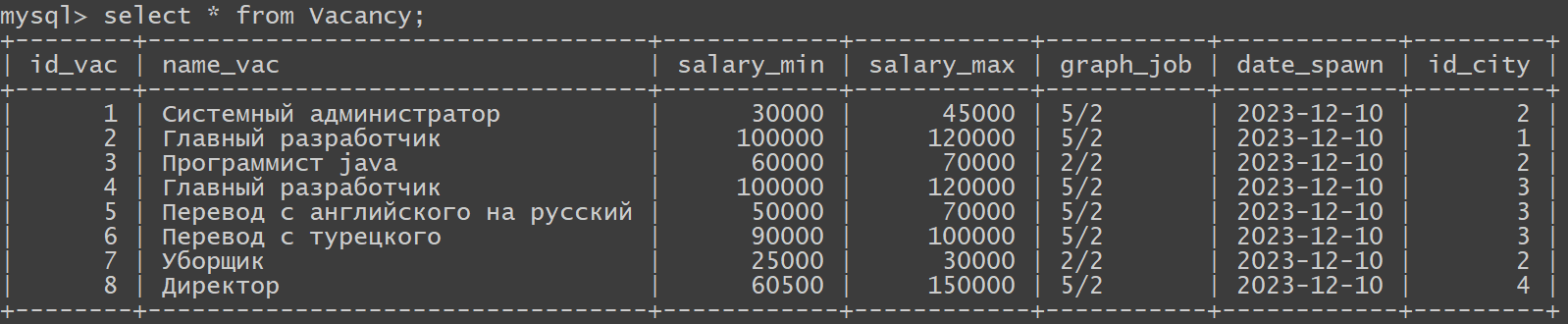
END;//

DELIMITER ;

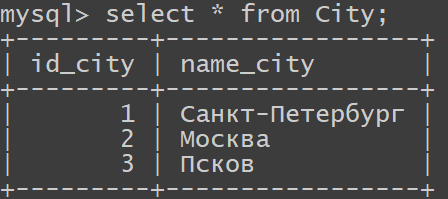
CALL del\_vac\_clear\_city(8);

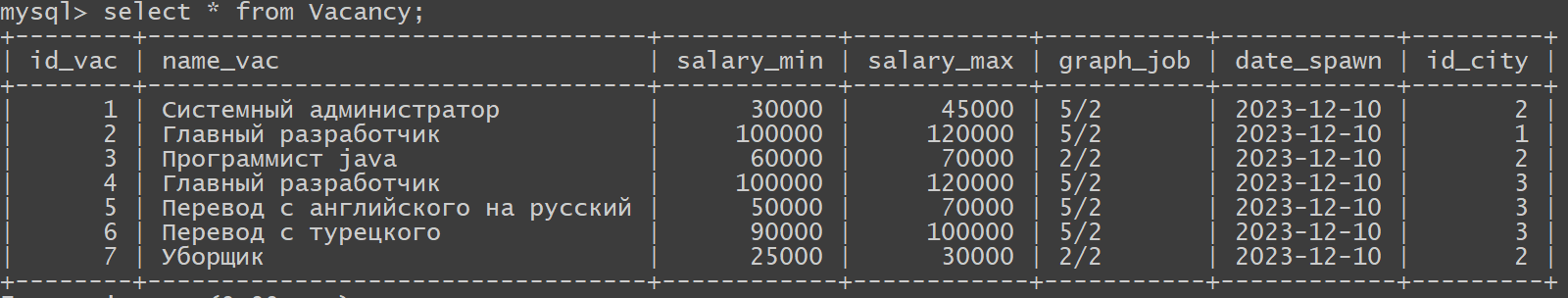
Наборы данных до вызова:





Наборы данных после вызова:

****

****

1. **Процедура каскадного удаления (При удалении города, удаляются связанные вакансии)**

DELIMITER //

CREATE PROCEDURE del\_city\_cascade (IN id\_city\_del INT)

BEGIN

Delete from Vacancy\_duty WHERE id\_vac in (SELECT id\_vac from Vacancy WHERE id\_city = id\_city\_del);

Delete from Vacancy\_education WHERE id\_vac in (SELECT id\_vac from Vacancy WHERE id\_city = id\_city\_del);

Delete from Vacancy\_requirement WHERE id\_vac in (SELECT id\_vac from Vacancy WHERE id\_city = id\_city\_del);

Delete from Vacancy\_employer WHERE id\_vac in (SELECT id\_vac from Vacancy WHERE id\_city = id\_city\_del);

Delete from Employer\_city WHERE id\_city = id\_city\_del;

Delete from Vacancy WHERE id\_city = id\_city\_del;

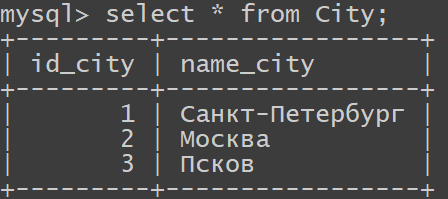
Delete from City WHERE id\_city = id\_city\_del;

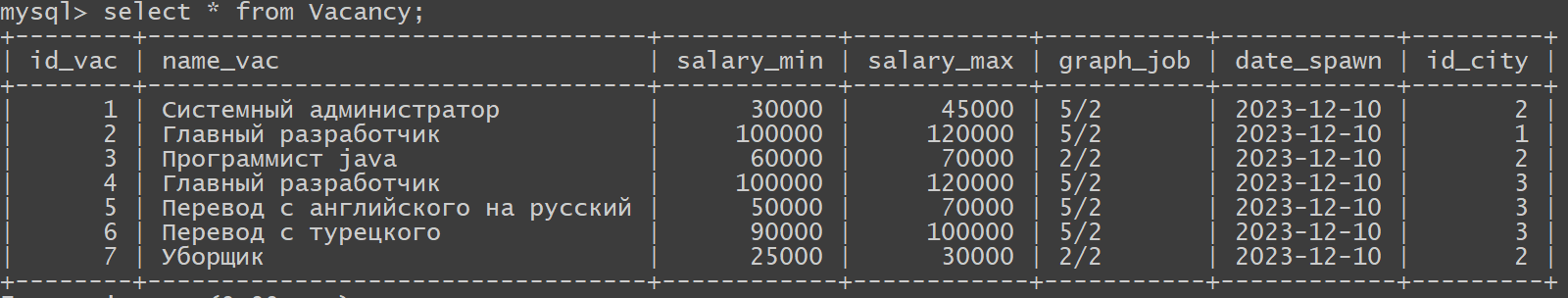
END;//

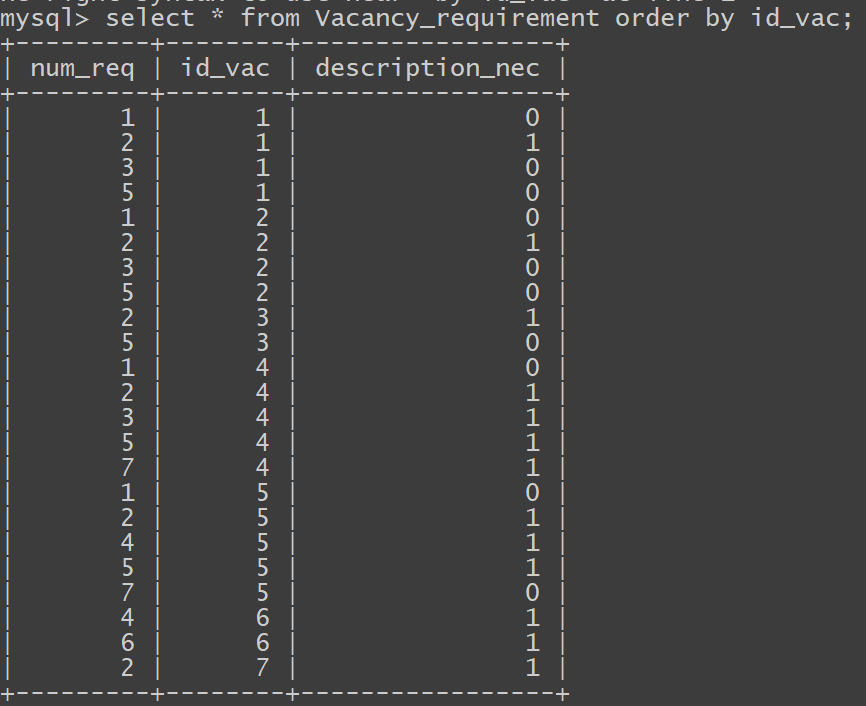
DELIMITER ;

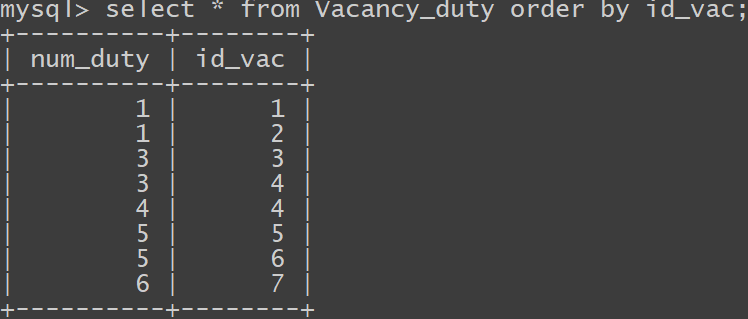
CALL del\_city\_cascade(3);

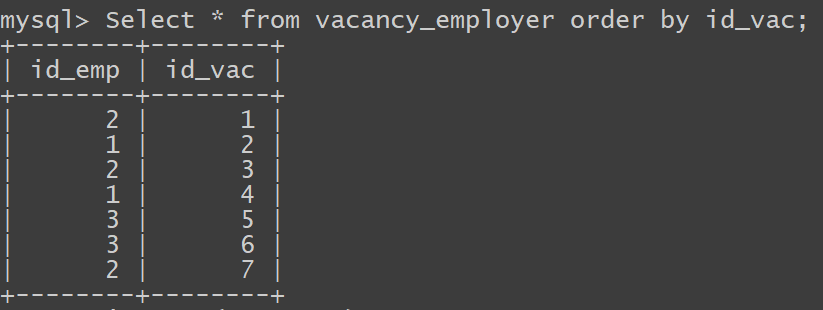
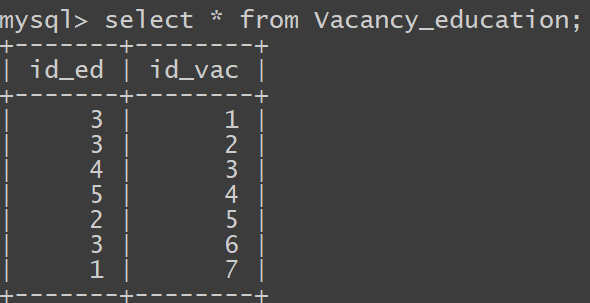
Наборы данных до вызова:

****

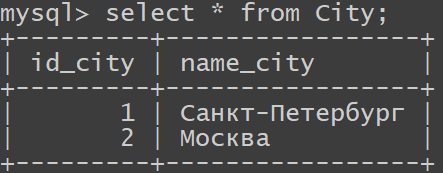
****

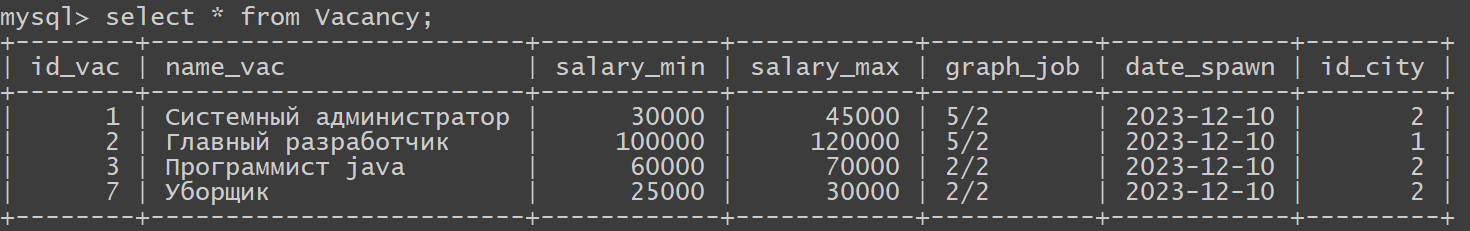
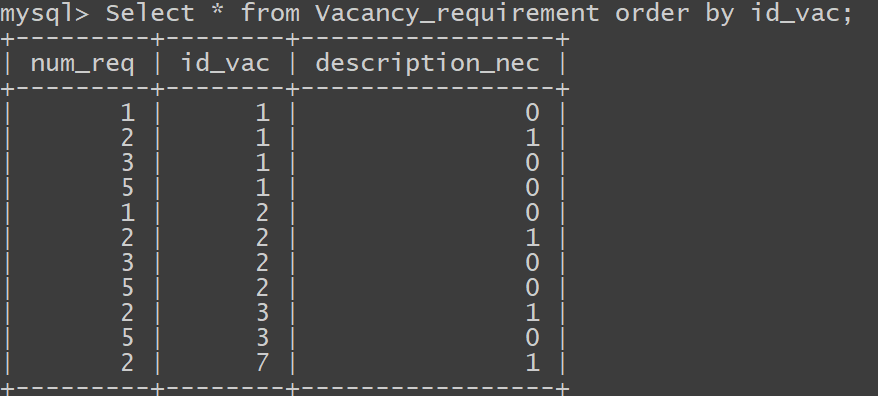
****

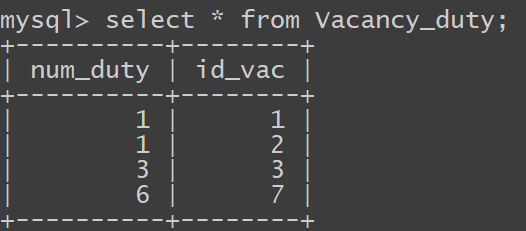
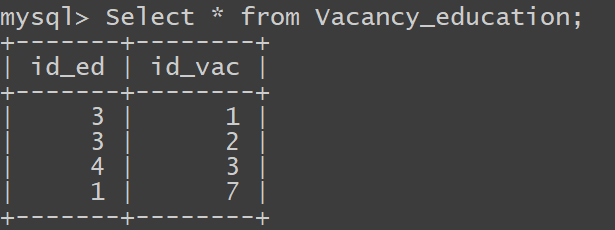
****

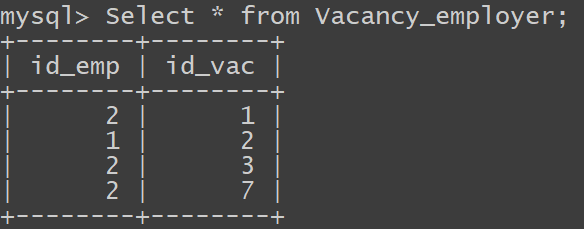
****

Наборы данных после вызова:



**** ****

**** ****

****

**Возврат количества вакансий:**

DELIMITER //

CREATE PROCEDURE count\_vac (out cnt\_vac INT)

BEGIN

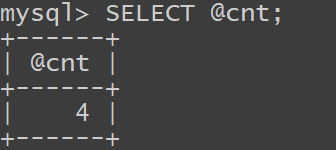
SELECT ifnull(count(id\_vac),0) into cnt\_vac from Vacancy;

END;//

DELIMITER ;

Call count\_vac(@cnt);

SELECT @cnt;



**Возврат количества вакансий - функция:**

DELIMITER //

CREATE FUNCTION fun\_count\_vac() returns int DETERMINISTIC

BEGIN

DECLARE cnt\_vac int DEFAULT 0;

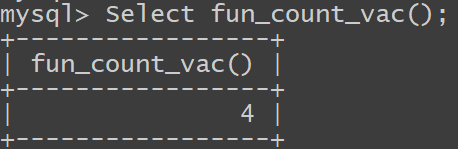
Set cnt\_vac=(SELECT ifnull(count(id\_vac),0) from Vacancy);

Return cnt\_vac;

END;//

DELIMITER ;

Select fun\_count\_vac();



**Формирование статистики во временной таблице**

DELIMITER //

CREATE PROCEDURE vac\_statistics ()

BEGIN

Create temporary table if not exists vac\_stat

(

id\_stat int auto\_increment primary key,

id\_vac int,

min\_salary int,

avg\_min\_salary int,

diff\_min\_avg\_salary int,

count\_req int,

count\_req\_avg double default 0,

diff\_cnt\_avg double default 0  
);

insert into vac\_stat (id\_vac, count\_req, min\_salary)

select Vacancy.id\_vac, count(num\_req) as count\_req, salary\_min from Vacancy

left join Vacancy\_requirement on Vacancy.id\_vac=Vacancy\_requirement.id\_vac group by Vacancy.id\_vac;

update vac\_stat set avg\_min\_salary=(select avg(salary\_min) from Vacancy);

update vac\_stat set diff\_min\_avg\_salary=min\_salary-avg\_min\_salary;

update vac\_stat set count\_req\_avg=(select avg(count\_req) from

(select Vacancy.id\_vac, count(num\_req) as count\_req from Vacancy

left join Vacancy\_requirement on Vacancy.id\_vac=Vacancy\_requirement.id\_vac group by Vacancy.id\_vac)q);

update vac\_stat set diff\_cnt\_avg=count\_req-count\_req\_avg;

select \* from vac\_stat;

select avg(diff\_min\_avg\_salary\*diff\_min\_avg\_salary) from vac\_stat;

drop table vac\_stat;

END;//

DELIMITER ;

Call vac\_statistics;

