**1) Связь "многие ко многим"**

1.1) Показать имена сотрудников, которые продали товар "Самсунг" и товар "Асус"

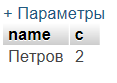
SELECT employees.name, COUNT(\*) as c

FROM ((employees JOIN orders USING(emp\_id)) JOIN merch USING(merch\_id))

WHERE merch.m\_name IN ("Асус", "Самсунг")

GROUP BY employees.name

HAVING c=2



1.2) Показать имена сотрудников, которые являются директорами магазинов из Москвы и Тулы.

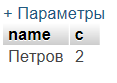
SELECT employees.name, COUNT(\*) as c

FROM (employees JOIN shops USING(emp\_id))

WHERE shops.adds IN ("Москва", "Тула")

GROUP BY employees.name

HAVING c=2



**2) Хранимые процедуры**

2.1) Создать хранимую процедуру Best\_Emp(…, …), которая определяет лучшего продавца по суммарному значению продаж за период дат.

DELIMITER //

CREATE PROCEDURE Best\_Emp(n1 DATE, n2 DATE)

BEGIN

SELECT sum(orders.amount\*merch.price) as s, employees.name

FROM employees

INNER JOIN orders ON orders.emp\_id = employees.emp\_id AND orders.data BETWEEN n1 AND n2

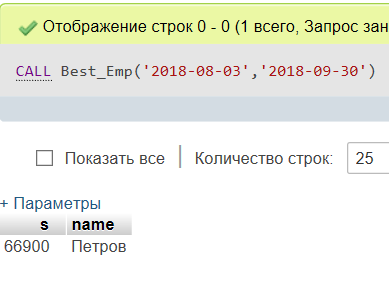
INNER JOIN merch ON orders.merch\_id = merch.merch\_id

GROUP by employees.name

ORDER BY s DESC

LIMIT 1;

END//   
DELIMITER ;



2.2) Создать хранимую процедуру List\_Merch(…), которая возвращает список  товаров, каждый из которых был проданы на сумму больше, чем заданное значение в параметре.

DELIMITER //

CREATE PROCEDURE List\_Merch(n int)

BEGIN

SELECT merch.m\_name, SUM(merch.price\*orders.amount) AS S

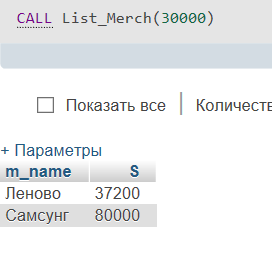
FROM (merch JOIN orders USING(merch\_id))

GROUP BY merch.m\_name

HAVING S>n;

END//

DELIMITER ;



2.3) Создать процедуру  (..., ...), которая покажет самый прибыльный магазин из 2 заданных городов.

DELIMITER //

CREATE PROCEDURE Best\_Shopp(a varchar(50), b varchar(50) )

BEGIN

SELECT SUM(orders.amount\*merch.price) AS S, shops.name

FROM shops

INNER JOIN orders ON orders.shop\_id=shops.shop\_id

INNER JOIN merch ON merch.merch\_id=orders.merch\_id

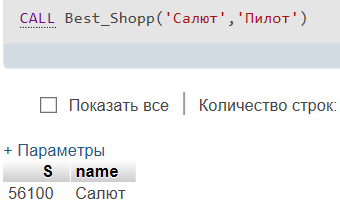
WHERE shops.name IN (a,b)

GROUP BY shops.name

ORDER BY S DESC LIMIT 1

END//

DELIMITER ;



2.4) Создать хранимую процедуру Itog(…) с одним входным параметром, которая выводит наименование товара по суммарному количеству проданных товаров:

DELIMITER //

CREATE PROCEDURE Itog(a varchar(50))

BEGIN

CASE a

WHEN 'Непопулярный товар' THEN SELECT merch.m\_name, SUM(orders.amount) as s FROM (merch JOIN orders USING (merch\_id)) GROUP BY merch.m\_name HAVING s<3;

WHEN 'Обычный товар' THEN SELECT merch.m\_name, SUM(orders.amount) as s FROM (merch JOIN orders USING (merch\_id)) GROUP BY merch.m\_name HAVING s BETWEEN 3 AND 5;

WHEN 'Популярный товар' THEN SELECT merch.m\_name, SUM(orders.amount) as s FROM (merch JOIN orders USING (merch\_id)) GROUP BY merch.m\_name HAVING s>5;

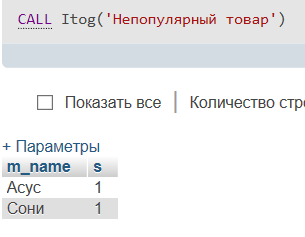
ELSE SELECT "error" as ERR;

END CASE;

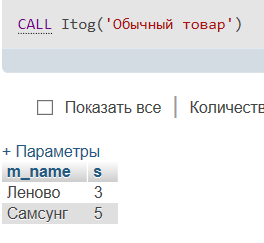
END//

DELIMITER ;

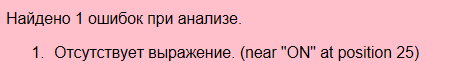
 а) Оценка 'Непопулярный товар', если число проданных товаров меньше 3



 б) Оценка 'Обычный товар', если число проданных товаров больше 3 и меньше 5



 с) Оценка 'Популярный товар', если число проданных товаров больше 5



**3) Триггеры**

3.1) Создать триггер Before\_Delete\_Merch, который при удалении товара удалял бы все записи о его продажах в таблице заказов.

DELIMITER //

CREATE TRIGGER Before\_Delete\_Merch

BEFORE DELETE

ON merch

FOR EACH ROW

BEGIN

DELETE FROM orders WHERE merch.merch\_id=orders.merch\_id;

END//

DELIMITER ;

3.2) Создать триггер Before\_Update\_Price, который увеличивает цену товара на 20% при ее изменении с помощью UPDATE.

DELIMITER //

CREATE TRIGGER Before\_Delete\_Merch

BEFORE UPDATE

ON merch

FOR EACH ROW

BEGIN

SET NEW.price=OLD.price\*1.2;

END//

DELIMITER ;