|  |  |  |
| --- | --- | --- |
| *БАЗЫ ДАННЫХ* | | |
| Кафедра САПР и ПК  Отчёт экзаменационного задания | ФИО студента | Белоусов Дмитрий Вадимович |
| Группа | ИВТ-260 |
| Дата выполнения | 15.06.2022 |
| ФИО преподавателя | Соколов Александр Александрович |
|  |

# Текст задания:

В цветочном магазине ведётся учёт цветов, букетов, заказов.

Для каждого цветка хранятся его наименование, размер по шкале от 1 до 10, цена, цвет.

Букет имеет сложность по шкале от 1 до 10, наценку за формирование букета, наименование и описание упаковки.

В букет входит множество цветов, один и тот же цветок может присутствовать в разных букетах.

У заказа хранится дата и время заказа, ФИО продавца.

В заказ может входить множество букетов, а букет может быть заказан множество раз.

Запросы:

1. Вывести все букеты заданной сложности и в заданной упаковке, список букетов: Наименование (string), Сложность (int), Упаковка (string).

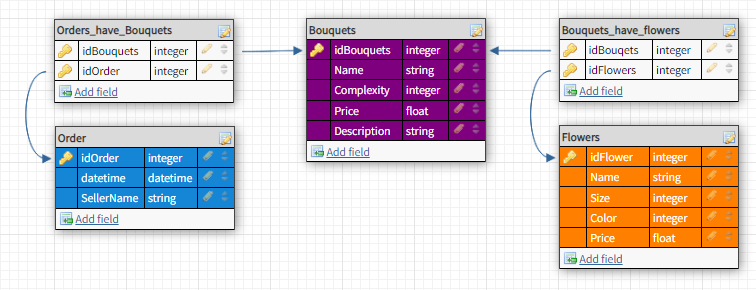
2. Вывести ТОП-5 всех цветов отсортированных по размеру в букетах заданной сложности, список всех цветов: Наименование (string), Размер (int), Цена (float), Цвет (string)

3. Вывести среднюю сложность букетов, сделанных продавцом с заданным ФИО за период с 01.01.2000 по 05.01.2000, число (float).

4. Вывести суммарную стоимость заказа от 08.01.2000 у продавца с заданным ФИО, число (float)

# Выполнение

## 1. Проектирование базы данных:



## 2. 1 Создание базы данных в MySQL

Для каждого цветка хранятся его наименование, размер по шкале от 1 до 10, цена, цвет.

CREATE SCHEMA IF NOT EXISTS flowerDB;

USE flowerDB ;

Для каждого цветка хранятся его наименование, размер по шкале от 1 до 10, цена, цвет.

CREATE TABLE IF NOT EXISTS flowers (

idFlowers INT NOT NULL AUTO\_INCREMENT,

Name VARCHAR(45) NULL DEFAULT NULL,

Size INT NULL DEFAULT NULL,

Color VARCHAR(50) DEFAULT NULL,

Price FLOAT DEFAULT NULL,

PRIMARY KEY (idFlowers)

);

Букет имеет сложность по шкале от 1 до 10, наценку за формирование букета, наименование и описание упаковки.

CREATE TABLE IF NOT EXISTS bouquets (

idBouquets INT NOT NULL AUTO\_INCREMENT,

Name VARCHAR(45) NULL DEFAULT NULL,

Description VARCHAR(100) NULL DEFAULT NULL,

Complexity INT DEFAULT NULL,

Price FLOAT DEFAULT NULL,

PRIMARY KEY (idBouquets)

);

У заказа хранится дата и время заказа, ФИО продавца.

CREATE TABLE IF NOT EXISTS orders (

idOrders INT NOT NULL AUTO\_INCREMENT,

SellerName VARCHAR(45) NULL DEFAULT NULL,

DateTime DATETIME NULL DEFAULT NULL,

PRIMARY KEY (idOrders)

);

В букет входит множество цветов, один и тот же цветок может присутствовать в разных букетах.

CREATE TABLE IF NOT EXISTS Bouquets\_have\_Flowers (

idBouquets INT NOT NULL,

idFlowers INT NOT NULL,

PRIMARY KEY (idBouquets, idFlowers),

INDEX idBouquetsInd (idBouquets ASC) VISIBLE,

INDEX idFlowersInd (idFlowers ASC) VISIBLE,

CONSTRAINT idBouquetsKey2

FOREIGN KEY (idBouquets)

REFERENCES bouquets (idBouquets),

CONSTRAINT idFlowersKey

FOREIGN KEY (idFlowers)

REFERENCES Flowers (idFlowers));

В заказ может входить множество букетов, а букет может быть заказан множество раз.

CREATE TABLE IF NOT EXISTS Orders\_have\_Bouquets (

idBouquets INT NOT NULL,

idOrders INT NOT NULL,

PRIMARY KEY (idBouquets, idOrders),

INDEX idBouquetsInd (idBouquets ASC) VISIBLE,

INDEX idOrdersInd (idOrders ASC) VISIBLE,

CONSTRAINT idBouquetsKey

FOREIGN KEY (idBouquets)

REFERENCES bouquets (idBouquets),

CONSTRAINT idOrdersKey

FOREIGN KEY (idOrders)

REFERENCES orders (idOrders));

## 2.2 Добавление данных в созданную базу данных:

-- INSERTING DATA

INSERT INTO orders (SellerName, DateTime) VALUES

('Diane', '2000-01-03 05:06'),

('Diane', '2000-01-12 08:59'),

('Diane', '2000-01-04 13:42'),

('Diane', '2000-01-02 14:50'),

('John', '2000-01-09 19:00'),

('John', '2000-02-25 15:00');

-- order for 2000-01-08

INSERT INTO orders (SellerName, DateTime) VALUES

('Diane', '2000-01-08 05:06'),

('Diane', '2000-01-08 08:59'),

('Diane', '2000-01-08 13:42'),

('Diane', '2000-01-08 14:50'),

('John', '2000-01-08 19:00'),

('Diane', '2000-01-08 05:06'),

('Diane', '2000-01-08 08:59'),

('Diane', '2000-01-08 13:42'),

('Diane', '2000-01-08 14:50'),

('John', '2000-01-08 19:00'),

('John', '2000-01-08 15:00');

INSERT INTO bouquets (Name, Description, Complexity, Price) VALUES

('Tenderness', 'fgvsefd', 5, 50),

('Passion', 'fdgvsreg', 6, 50),

('Luxury', 'fzvdzvsfdzv', 9, 50),

('Rapture', 'dhydtby', 7, 50),

('Sadness', 'dsnbyhdff', 6, 50);

INSERT INTO flowers (Name, Size, Color, Price) VALUES

('rose white', 4, 'white', 200),

('rose black', 4, 'black', 300),

('rose red', 4, 'red', 150),

('ehrysanthemum', 2, 'pink', 100),

('eustoma', 3, 'white', 200),

('lilies', 2, 'yellow', 150);

INSERT INTO orders\_have\_bouquets (idBouquets, idOrders) VALUES

(2, 1),

(5, 2),

(4, 2),

(1, 3),

(3, 4),

(5, 4),

(1, 5),

(3, 6);

-- orders for 2000-01-08

INSERT INTO orders\_have\_bouquets (idBouquets, idOrders) VALUES

(1, 7),

(5, 8),

(4, 8),

(4, 9),

(1, 10),

(3, 11),

(5, 11),

(5, 12),

(1, 13),

(2, 14),

(1, 15),

(5, 16),

(3, 17);

INSERT INTO bouquets\_have\_flowers (idBouquets, idFlowers) VALUES

(1, 1),

(1, 4),

(1, 5),

(2, 4),

(2, 3),

(3, 3),

(3, 5),

(4, 1),

(4, 5),

(5, 2);

## 3. Создание запросов:

1. Вывести все букеты заданной сложности и в заданной упаковке, список букетов: Наименование (string), Сложность (int), Упаковка (string).

-- 1. Output all bouquets with given Complexity

SELECT Name, Complexity, Price FROM bouquets

WHERE Complexity = 5;



2. Вывести ТОП-5 всех цветов отсортированных по размеру в букетах заданной сложности, список всех цветов:

Наименование (string), Размер (int), Цена (float), Цвет (string)

-- 2. Output TOP-5 of all flowers, sorted by size in bouquets with given complexity

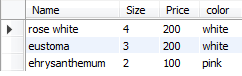
SELECT flowers.Name, Size, flowers.Price, flowers.color FROM flowers

INNER JOIN bouquets\_have\_flowers ON flowers.idFlowers = bouquets\_have\_flowers.idFlowers

INNER JOIN bouquets ON bouquets.idBouquets = bouquets\_have\_flowers.idBouquets

WHERE Complexity = 5

ORDER BY Size DESC LIMIT 5;



3. Вывести среднюю сложность букетов, сделанных продавцом с заданным ФИО за период с 01.01.2000 по 05.01.2000, число (float).

-- 3. Output average complexity of bouquets, made by given seller BETWEEN 01.01.2000 AND 05.01.2000

SELECT orders.SellerName, AVG(bouquets.Complexity) as Average\_complexity FROM orders

INNER JOIN orders\_have\_bouquets ON orders.idOrders = orders\_have\_bouquets.idOrders

INNER JOIN bouquets ON bouquets.idBouquets = orders\_have\_bouquets.idBouquets

WHERE orders.SellerName = "Diane" AND orders.DateTime BETWEEN "2000-01-01 00:00" AND "2000-01-05 23:59"

GROUP BY orders.SellerName;



4. Вывести суммарную стоимость заказа от 08.01.2000 у продавца с заданным ФИО, число (float)

-- 4. Sum Price of orders of given seller in 08.01.2000

SELECT sum\_price.SellerName, SUM(sum\_price.Price) as 'Sum for a day' FROM

(SELECT orders.SellerName, orders.idOrders, SUM(flowers.Price) as Price FROM orders -- sum price of flowers for every order

INNER JOIN orders\_have\_bouquets ON orders.idOrders = orders\_have\_bouquets.idOrders

INNER JOIN bouquets ON bouquets.idBouquets = orders\_have\_bouquets.idBouquets

INNER JOIN bouquets\_have\_flowers ON bouquets.idBouquets = bouquets\_have\_flowers.idFlowers

INNER JOIN flowers ON flowers.idFlowers = bouquets\_have\_flowers.idBouquets

WHERE SellerName = 'Diane' AND orders.DateTime BETWEEN "2000-01-08 00:00" AND "2000-01-08 23:59"

GROUP BY orders.idOrders

UNION ALL

SELECT orders.SellerName, orders.idOrders, SUM(bouquets.Price) as Price FROM orders -- sum price of packing for every order

INNER JOIN orders\_have\_bouquets ON orders.idOrders = orders\_have\_bouquets.idOrders

INNER JOIN bouquets ON bouquets.idBouquets = orders\_have\_bouquets.idBouquets

INNER JOIN bouquets\_have\_flowers ON bouquets.idBouquets = bouquets\_have\_flowers.idFlowers

INNER JOIN flowers ON flowers.idFlowers = bouquets\_have\_flowers.idBouquets

WHERE SellerName = 'Diane' AND orders.DateTime BETWEEN "2000-01-08 00:00" AND "2000-01-08 23:59"

GROUP BY orders.idOrders) as sum\_price

GROUP BY sum\_price.SellerName;

