Тестовое задание

Выполнил: Насирли Самир Ровна оглы

<https://balashiha.hh.ru/resume/9d568852ff0cfc96080039ed1f6d6c7a367869>

Тестовое задание:

Возьмите свою страницу Вконтакте, соберите по ней таблицу с датой постов и количеством лайков и напишите SQL-запросы, которые позволят ответить на вопрос: что больше всего влияет на количество лайков: время суток публикации, день недели или промежуток между постами. В качестве ответа приложите таблицу в формате CSV, SQL-запросы с необходимыми пояснениями и выводом. Вместо своей страницы можно взять станицу друга, в любом случае укажите адрес страницы.

Была создана таблица:

CREATE TABLE posts (

post\_id INT AUTO\_INCREMENT PRIMARY KEY,

post\_date DATETIME NOT NULL,

post\_date\_last DATETIME NOT NULL,

likes\_count INT NOT NULL,

day\_of\_week ENUM('Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'),

time\_of\_day ENUM('morning', 'afternoon', 'evening', 'night'),

time\_since\_last\_post TIME,

time\_type ENUM('from 1 to 5 minutes', 'from 5 minutes to an hour', 'from one to 5 hours', 'from 5 to 12 hours', 'from 12 to 24 hours', 'More than a day')

);

Созданы триггеры для автоматического заполнения некоторых атрибутов.

DELIMITER //

CREATE TRIGGER calculate\_time\_since\_last\_post

BEFORE INSERT ON posts

FOR EACH ROW

BEGIN

-- Вычисляем разницу между post\_date и post\_date\_last

SET NEW.time\_since\_last\_post = TIMEDIFF(NEW.post\_date, NEW.post\_date\_last);

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER set\_day\_of\_week

BEFORE INSERT ON posts

FOR EACH ROW

BEGIN

-- Определяем день недели по post\_date

SET NEW.day\_of\_week = DAYNAME(NEW.post\_date);

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER set\_time\_of\_day

BEFORE INSERT ON posts

FOR EACH ROW

BEGIN

-- Определяем время суток по post\_date

DECLARE hour\_of\_day INT;

SET hour\_of\_day = HOUR(NEW.post\_date);

-- Записываем время суток в поле time\_of\_day

IF hour\_of\_day >= 6 AND hour\_of\_day < 12 THEN

SET NEW.time\_of\_day = 'morning';

ELSEIF hour\_of\_day >= 12 AND hour\_of\_day < 18 THEN

SET NEW.time\_of\_day = 'afternoon';

ELSEIF hour\_of\_day >= 18 AND hour\_of\_day < 24 THEN

SET NEW.time\_of\_day = 'evening';

ELSE

SET NEW.time\_of\_day = 'night';

END IF;

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER set\_time\_type

BEFORE INSERT ON posts

FOR EACH ROW

BEGIN

DECLARE time\_diff TIME;

DECLARE time\_type\_value ENUM('from 1 to 5 minutes', 'from 5 minutes to an hour', 'from one to 5 hours', 'from 5 to 12 hours', 'from 12 to 24 hours', 'More than a day');

SET time\_diff = NEW.time\_since\_last\_post;

IF time\_diff <= '00:05:00' THEN

SET time\_type\_value = 'from 1 to 5 minutes';

ELSEIF time\_diff > '00:05:00' AND time\_diff <= '01:00:00' THEN

SET time\_type\_value = 'from 5 minutes to an hour';

ELSEIF time\_diff > '01:00:00' AND time\_diff <= '05:00:00' THEN

SET time\_type\_value = 'from one to 5 hours';

ELSEIF time\_diff > '05:00:00' AND time\_diff <= '12:00:00' THEN

SET time\_type\_value = 'from 5 to 12 hours';

ELSEIF time\_diff > '12:00:00' AND time\_diff <= '24:00:00' THEN

SET time\_type\_value = 'from 12 to 24 hours';

ELSE

SET time\_type\_value = 'More than a day';

END IF;

SET NEW.time\_type = time\_type\_value;

END //

DELIMITER ;

Данные были импортированы из файла Данные.csv. Исходные данные находятся в файле Полные данные.csv.

Изначально были написаны простые отдельные запросы по каждой из категорий.

SELECT

time\_of\_day,

AVG(likes\_count) AS avg\_likes

FROM

posts

GROUP BY

time\_of\_day

ORDER BY

avg\_likes DESC;

SELECT

day\_of\_week,

AVG(likes\_count) AS avg\_likes

FROM

posts

GROUP BY

day\_of\_week

ORDER BY

avg\_likes DESC;

SELECT

time\_type,

AVG(likes\_count) AS avg\_likes

FROM

posts

GROUP BY

time\_type

ORDER BY

avg\_likes DESC;

Потом был написан общий запрос. Результат которого находится в файле Результат.csv

SELECT

time\_of\_day,

AVG(likes\_count) AS avg\_likes

FROM

posts

GROUP BY

time\_of\_day

ORDER BY

avg\_likes DESC;

SELECT

day\_of\_week,

AVG(likes\_count) AS avg\_likes

FROM

posts

GROUP BY

day\_of\_week

ORDER BY

avg\_likes DESC;

SELECT

time\_type,

AVG(likes\_count) AS avg\_likes

FROM

posts

GROUP BY

time\_type

ORDER BY

avg\_likes DESC;