**Коротко аналитические данные. Подробности – в файле ниже, в Excel-графиках, а совсем подробно в SQL-скрипте**

* период данных - 2016-й и 2017-й года
* изначально количество строк = 2002752
* после обработки данных строк осталось = 1999462
* уникальных пользователей 1014352
* средняя годовая выручка. Для 2016 г. это 4929280, а для 2017 г. 7500387, выросли аж на 52% от показателей прошлого года
* в среднем суточный чек по годам, в 2016-м это 2101 денежных знаков, а в 2017-м 2400, суточный чек за год вырос на 14%
* средний чек по месяцам - здесь видим тренд на полный вперёд и радуемся этому, хлопаем в ладоши
* показатель роста среднего чека по месяцам добавил в Excel и построил по него симпатичный график
* среднее количество заказов на пользователя 1,97 штук
* пользователей, переставших покупать в 2017-м году, 359605 уникальных голов, что составляет в среднем 35% от общего количества уникальных пользователей
* ID самого активного по кол-ву покупок пользователя –id=765861, у него 3182 заказа
* коэффициенты сезонности по месяцам, сделать предположение об аномалии (если она есть) - решение в таблице в Excel, там же график среднего чека по месяцам
* Задание построить график вероятности второй покупки по дням сразу после первой - для примера выборки взял 100 покупателей, совершивших от 2-х до 10-ти заказов включительно. Есть графики "падение вероятности 2+ покупки, данные по количеству пользователей" и "рост кол-ва дней от 2-й до последующих покупок"
* Дополнительно собрал таблицу из выборки в 239524 шт. клиентов, совершавших от 2-ч до 50-ти заказов, и посчитал у них время между первым и вторым заказом. Получилось так:

🡪 Минимальное время 0 дней

🡪 Максимальное время 727 дня

🡪 Среднее время 72,5 дня, т.е. 2 с хвостом месяца

**1. Залейте в свою БД данные по продажам:**

-- создаём пустую таблицу

**DROP** **TABLE** **IF** **EXISTS** orders\_20190822;

**CREATE** **TABLE** orders\_20190822(

id\_o **INT**,

user\_id **INT**,

price **VARCHAR**(50),

o\_date **VARCHAR**(50));

-- зальём в таблицу данные из csv-файла

**LOAD** **DATA** **INFILE** 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/orders\_20190822.csv'

**INTO** **TABLE** orders\_20190822

FIELDS **TERMINATED** **BY** ';'

**ENCLOSED** **BY** '"'

**LINES** **TERMINATED** **BY** '\r\n'

**IGNORE** 1 **ROWS**;

-- проверим заполнение и количество строк

**SELECT** \* **FROM** orders\_20190822;

**SELECT** **COUNT**(\*) **FROM** orders\_20190822;

-- проверим количество пропусков и сразу, если что, грохнем их (52 строчки)

**SELECT** \* **FROM** orders\_20190822 **WHERE** id\_o = '' **OR** id\_o = **NULL** **OR** user\_id = '' **OR** user\_id = **NULL** **OR** price **LIKE** '0,%' **OR** o\_date = '' **OR** o\_date = **NULL**;

**DELETE** **FROM** orders\_20190822 **WHERE** id\_o = '' **OR** id\_o = **NULL** **OR** user\_id = **NULL** **OR** price **LIKE** '0,%' **OR** o\_date = '' **OR** o\_date = **NULL**;

-- в получившейся таблице цена с запятой и типа VARCHAR и дата в виде строки - исправим эти столбцы на FLOAT и DATA, соответственно

-- как? создадим такой же каркас для второй таблицы, только price в ней будет типа FLOAT, а o\_date типа DATE

-- и в процедуре пройдёмся курсором по старой таблице, заполняя её данными новую таблицу, но уже с отформатированными значениеми price и o\_data

**DROP** **TABLE** **IF** **EXISTS** orders\_table\_price\_float;

**CREATE** **TABLE** orders\_table\_price\_float(

id\_o **INT**,

user\_id **INT**,

price **FLOAT**,

o\_date **DATE**);

**DELIMITER $$**

**DROP** **PROCEDURE** **IF** **EXISTS** copy\_db\_orders$$

**CREATE** **PROCEDURE** copy\_db\_orders()

**BEGIN**

**DECLARE** id\_o, user\_id **INT**;

**DECLARE** price, o\_date **VARCHAR**(50);

**DECLARE** is\_end **INT** **DEFAULT** 0;

**DECLARE** curcat **CURSOR** **FOR** **SELECT** \* **FROM** orders\_20190822;

**DECLARE** **CONTINUE** HANDLER **FOR** **NOT** **FOUND** **SET** is\_end = 1;

**OPEN** curcat;

cyc: **LOOP**

**FETCH** curcat **INTO** id\_o, user\_id, price, o\_date;

**IF** is\_end = 1 **THEN**

**LEAVE** cyc;

**END** **IF**;

**INSERT** **IGNORE** **INTO** orders\_table\_price\_float **VALUES**(

id\_o,

user\_id,

**ROUND**(**CONVERT**((**REPLACE**(price, ',', '.')), **FLOAT**), 2),

STR\_TO\_DATE(o\_date, '%d.%m.%Y'));

**END** **LOOP** cyc;

**CLOSE** curcat;

**END**$$

**DELIMITER ;**

-- проверим

**SELECT** \* **FROM** orders\_table\_price\_float **ORDER** **BY** price **DESC**;

**SELECT** (**SELECT** **COUNT**(\*) **FROM** orders\_20190822) **AS** orders\_20190822, (**SELECT** **COUNT**(\*) **FROM** orders\_table\_price\_float) **AS** orders\_table\_price\_float;

-- на всякий случай, чтоб не повредить полученную новую таблицу, которая формировалась час с лишним, работать будем с её копией. Создадим копию

**DROP** **TABLE** **IF** **EXISTS** copy\_orders\_table\_price\_float;

**CREATE** **TABLE** copy\_orders\_table\_price\_float **SELECT** \* **FROM** orders\_table\_price\_float;

-- пошли по колонкам данные изучать по всякому:

-- PRICE

-- смотрим минимальный price

**SELECT** price **FROM** copy\_orders\_table\_price\_float **ORDER** **BY** price;

-- есть отрицательные цены, как так? Удалим их и все, что ценой меньше 50-ти

**DELETE** **FROM** copy\_orders\_table\_price\_float cotpf **WHERE** price < 50

-- смотрим максимальный price

**SELECT** price **FROM** copy\_orders\_table\_price\_float cotpf **ORDER** **BY** price **DESC**;

-- ну один самый крупный снесём, уж больно он выбивается из всех остальных, похож на выброс. Остальные крупные похожи на b2b-заказы, что допустимо

**DELETE** **FROM** copy\_orders\_table\_price\_float cotpf **ORDER** **BY** price **DESC** **LIMIT** 1;

-- USER\_ID

-- проверим, нет ли частых повторений у user\_id - ну вот есть один user\_id=765861 с повторами 3182 раза.

-- но на 2 с хвостом миллиона строк это 0,15%, так что будем считать, что норм, просто активный покупатель

**DESCRIBE** copy\_orders\_table\_price\_float;

**SELECT** **COUNT**(user\_id) **AS** `count`, user\_id **FROM** copy\_orders\_table\_price\_float cotpf **GROUP** **BY** user\_id **ORDER** **BY** `count` **DESC**;

-- ID\_O

-- смотрим, нет ли одинаковых id у заказов, ура, нет:

**SELECT** **COUNT**(\*), **COUNT**(**DISTINCT** id\_o) **FROM** copy\_orders\_table\_price\_float;

-- O\_DATE

-- мы свойствах файла orders\_20190822.csv можем посмотреть дату последнего изменения файла, вот так:

-- \! for %x in ('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/orders\_20190822.csv') do (echo %~nx %~tx)

-- C:\ProgramData\MySQL\MySQL Server 8.0\Uploads>(echo orders\_20190822 22.08.2019 07:38 )

-- orders\_20190822 22.08.2019 07:38

-- получается, что все даты выше 22.08.2019 г. будут некорректные. Посмотрим, есть ли у нас такие (спойлер - нет таких)

**SELECT** \* **FROM** copy\_orders\_table\_price\_float **WHERE** o\_date >= '2019-08-22';

-- а что с нижней планкой? Есть ли выбросы и левые года? Проверяем количество строк в году - всё норм

**SELECT** **YEAR**(o\_date) **AS** `year`, **COUNT**(\*) **AS** `count` **FROM** copy\_orders\_table\_price\_float **GROUP** **BY** `year`;

-- посмотрим на количество заказов по дням, отсортировав их по возрастанию. Так мы можем прикинуть, нет ли "странностей" в данных

**SELECT** **CONCAT**(**DAYOFMONTH**(o\_date), ', ',

**CASE**

**WHEN** **MONTH**(o\_date) = 1 **THEN** 'январь'

**WHEN** **MONTH**(o\_date) = 2 **THEN** 'февраль'

**WHEN** **MONTH**(o\_date) = 3 **THEN** 'март'

**WHEN** **MONTH**(o\_date) = 4 **THEN** 'апрель'

**WHEN** **MONTH**(o\_date) = 5 **THEN** 'май'

**WHEN** **MONTH**(o\_date) = 6 **THEN** 'июнь'

**WHEN** **MONTH**(o\_date) = 7 **THEN** 'июль'

**WHEN** **MONTH**(o\_date) = 8 **THEN** 'август'

**WHEN** **MONTH**(o\_date) = 9 **THEN** 'сентябрь'

**WHEN** **MONTH**(o\_date) = 10 **THEN** 'октябрь'

**WHEN** **MONTH**(o\_date) = 11 **THEN** 'ноябрь'

**ELSE** 'декабрь'

**END**, ', ',

**YEAR**(o\_date), ' ', 'год'

) **AS** `date`, **COUNT**(\*) **FROM** copy\_orders\_table\_price\_float **GROUP** **BY** o\_date **ORDER** **BY** **COUNT**(\*);

-- ожидаемо "провальные" по заказам дни - это дни перед длинными праздниками, а "хорошие" дни - вторая половина IV-го квартала, похоже на правду

-- данные почистили, более-менее проверили, приступаем к остальным этапам дз:

**2. Проанализировать, какой период данных выгружен**

**SELECT** **MONTH**(o\_date) **AS** `month`, **YEAR**(o\_date) **AS** `year` **FROM** copy\_orders\_table\_price\_float **GROUP** **BY** `month`, `year`;

-- период данных - 2016-й и 2017-й года

**3. Посчитать кол-во строк, кол-во заказов и кол-во уникальных пользователей, кот совершали заказы.**

-- изначально количество строк = 2002752

**SELECT** **COUNT**(\*) **FROM** orders\_20190822;

-- после обработки данных строк осталось = 1999462

-- уникальных пользователей 1014352

**SELECT** **COUNT**(\*) **FROM** copy\_orders\_table\_price\_float;

**SELECT**

(**SELECT** **COUNT**(id\_o) **FROM** copy\_orders\_table\_price\_float) **AS** `количество заказов`,

(**SELECT** **COUNT**(**DISTINCT** user\_id) **FROM** copy\_orders\_table\_price\_float) **AS** `количество уникальных пользователей`;

**4. По годам и месяцам посчитать средний чек, среднее кол-во заказов на пользователя, сделать вывод , как изменялись это показатели Год от года.**

-- общую сумму по месяцам посмотреть можно так:

**SELECT**

**YEAR**(o\_date) **AS** `year`,

**MONTH**(o\_date) **AS** `month`,

**ROUND**(**SUM**(price)) **AS** `sum`

**FROM** copy\_orders\_table\_price\_float

**GROUP** **BY** `year`, `month`;

-- а вот средняя годовая выручка. Для 2016 г. это 4929280, а для 2017 г. 7500387, выросли аж на 52% от показателей прошлого года

**SELECT**

**ROUND**((**SELECT** **ROUND**(**SUM**(price)) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2016)/(**SELECT** **COUNT**(**DISTINCT** o\_date) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2016))

**AS** `дневная выручка в 2016`,

**ROUND**((**SELECT** **ROUND**(**SUM**(price)) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2017)/(**SELECT** **COUNT**(**DISTINCT** o\_date) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2017))

**AS** `дневная выручка в 2017`;

-- в среднем суточный чек по годам, в 2016-м это 2101 денежных знаков, а в 2017-м 2400, суточный чек за год вырос на 14%

**SELECT**

**ROUND**((**SELECT** **ROUND**(**SUM**(price)) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2016)/(**SELECT** **COUNT**(**DISTINCT** id\_o) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2016))

**AS** `средний чек 2016`,

**ROUND**((**SELECT** **ROUND**(**SUM**(price)) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2017)/(**SELECT** **COUNT**(**DISTINCT** id\_o) **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2017))

**AS** `средний чек 2017`;

-- средний чек по месяцам, здесь видим тренд на полный вперёд и радуемся этому, хлопаем в ладоши

-- показатель роста среднего чека по месяцам добавил в Excel и построил по него симпатичный график

**SELECT**

**YEAR**(o\_date) **AS** `year`,

**MONTH**(o\_date) **AS** `month`,

**ROUND**(**AVG**(price)) **AS** `sum`

**FROM** copy\_orders\_table\_price\_float

**GROUP** **BY** `year`, `month`;

-- всего заказов у каждого пользователя:

**SELECT** user\_id, **COUNT**(user\_id) **AS** `count` **FROM** copy\_orders\_table\_price\_float **GROUP** **BY** user\_id;

-- среднее количество заказов на пользователя 1,97 штук

**SET** @unique\_users := (**SELECT** **COUNT**(\*) **FROM** (**SELECT** **COUNT**(user\_id) **FROM** copy\_orders\_table\_price\_float **GROUP** **BY** user\_id) **AS** c);

**SET** @all\_orders := (**SELECT** **COUNT**(id\_o) **FROM** copy\_orders\_table\_price\_float);

**SELECT** @all\_orders **AS** 'всего заказов', @unique\_users **AS** 'уникальных покупателей', **ROUND**(@all\_orders/@unique\_users, 2) **AS** `среднее кол-во заказов на одного покупателя`;

**5. Найти кол-во пользователей, кот покупали в одном году и перестали покупать в следующем.**

-- таких пользователей 359605 уникальных голов, что сочтавляет в среднем 35% от общего количества уникальных пользователей

**SET** @users\_2016 := (**SELECT** **COUNT**(**DISTINCT** user\_id) **FROM** copy\_orders\_table\_price\_float **WHERE** user\_id **NOT** **IN** (**SELECT** user\_id **FROM** copy\_orders\_table\_price\_float **WHERE** **YEAR**(o\_date) = 2017));

**SELECT** @users\_2016 **AS** `покупатели 2016 г., которых нет в 2017-м`, (**ROUND**(@users\_2016/@unique\_users\*100)) **AS** `это % от общего кол-ва уникальных покупателей`;

**6. Найти ID самого активного по кол-ву покупок пользователя.**

-- это пользователь с id=765861, у него 3182 заказа

**SELECT** user\_id **AS** `id покупателя-красавчика`, **COUNT**(user\_id) **AS** `всего заказов у него` **FROM** copy\_orders\_table\_price\_float cotpf **GROUP** **BY** user\_id **ORDER** **BY** `всего заказов у него` **DESC** **LIMIT** 1;

**7. Найти коэффициенты сезонности по месяцам, сделать предположение об аномалии (если она есть)**

-- **решение в таблице в Excel**, там же график среднего чека по месяцам

-- а данные для Excel взяты так:

**SELECT**

**YEAR**(o\_date) **AS** `year`,

**MONTH**(o\_date) **AS** `month`,

**ROUND**(**SUM**(price)) **AS** `sum`

**FROM** copy\_orders\_table\_price\_float

**GROUP** **BY** `year`, `month`;

**8. Построить график вероятности второй покупки по дням сразу после первой**

-- для примера выборки взял 100 покупателей, совершивших от 2-х до 10-ти заказов включительно

-- **для графика полученные данные по ним перенесём в Excel, там и построим**

-- "график падения вероятности 2+ покупки, данные по количеству пользователей" и

-- "график роста среднего кол-во дней от 2-й до 10-й покупки"

**SELECT**

user\_id,

**COUNT**(user\_id) **AS** `count`,

**GROUP\_CONCAT**(

o\_date

**ORDER** **BY** o\_date

**SEPARATOR** ', ')

**AS** `orders dates`

**FROM** copy\_orders\_table\_price\_float

**GROUP** **BY** user\_id

**HAVING** `count` >= 2 **AND** `count` <= 10

**LIMIT** 100;

-- для получения кол-ва дней между первым и вторым заказом проделаем вот что:

-- создадим временную таблицу, в которую положим только user\_id и даты заказов

-- сделаем ограничение в выборке клиентов по заказам, добавили только тех, кто сделал более 1-го заказа, но менее 50-ти, таких получится 239524 шт, более чем достаточно для объективной выборки

-- далее в процедуре курсором загоним эти данные в новую таблицу days\_between\_first\_and\_second\_order, положив в неё user\_id и разницу в днях между первым и вторым заказом

-- грузится где-то полчаса. Если не хочется ждать, то можно взять выборку поменьше

**DROP** **TEMPORARY** **TABLE** **IF** **EXISTS** sample\_of\_buyers;

**CREATE** **TEMPORARY** **TABLE** sample\_of\_buyers

**SELECT**

user\_id,

**GROUP\_CONCAT**(

o\_date

**ORDER** **BY** o\_date

**SEPARATOR** ', ')

**AS** `orders dates`

**FROM** copy\_orders\_table\_price\_float

**GROUP** **BY** user\_id

**HAVING** **COUNT**(user\_id) > 1 **AND** **COUNT**(user\_id) < 50;

**SELECT** \* **FROM** sample\_of\_buyers;

**SELECT** **COUNT**(\*) **FROM** sample\_of\_buyers;

**DESCRIBE** sample\_of\_buyers;

**DROP** **TABLE** **IF** **EXISTS** days\_between\_first\_and\_second\_order;

**CREATE** **TABLE** days\_between\_first\_and\_second\_order (user\_id **INT**, difference\_of\_days **INT**);

**DELIMITER $$**

**DROP** **PROCEDURE** **IF** **EXISTS** insert\_to\_days\_between\_first\_and\_second\_order$$

**CREATE** **PROCEDURE** insert\_to\_days\_between\_first\_and\_second\_order()

**BEGIN**

**DECLARE** user\_id **INT**;

**DECLARE** `orders dates` **TEXT**;

**DECLARE** is\_end **INT** **DEFAULT** 0;

**DECLARE** curcat **CURSOR** **FOR** **SELECT** \* **FROM** sample\_of\_buyers;

**DECLARE** **CONTINUE** HANDLER **FOR** **NOT** **FOUND** **SET** is\_end = 1;

**OPEN** curcat;

cyc: **LOOP**

**FETCH** curcat **INTO** user\_id, `orders dates`;

**IF** is\_end = 1 **THEN**

**LEAVE** cyc;

**END** **IF**;

**INSERT** **IGNORE** **INTO** days\_between\_first\_and\_second\_order **VALUES**(

user\_id,

TIMESTAMPDIFF(

**DAY**,

**SUBSTRING\_INDEX**(`orders dates`, ',', 1),

**SUBSTRING\_INDEX**(**SUBSTRING\_INDEX**(`orders dates`, ',', 2), ',', -1)));

**END** **LOOP** cyc;

**CLOSE** curcat;

**END**$$

**DELIMITER ;**

**CALL** insert\_to\_days\_between\_first\_and\_second\_order;

-- готово

**SELECT** \* **FROM** days\_between\_first\_and\_second\_order;

**SELECT** **COUNT**(\*) **FROM** days\_between\_first\_and\_second\_order;

-- ну и вот статистика выборки: самое малое время между 1-м и 2-м заказом = 0, самое долгое = 727 дней и среднее время = 72.5 дня

**SELECT**

**MIN**(difference\_of\_days) **AS** `минимальное время`,

**MAX**(difference\_of\_days) **AS** `максимальное время`,

**AVG**(difference\_of\_days) **AS** `среднее время`

**FROM** days\_between\_first\_and\_second\_order;