**Лев Хаак**

Финальный проект по учебному курсу

**«SQL для аналитиков»**

**(Международная школа профессий, преподаватель Павел Долганов).**

***Задача:*** *спроектировать, создать и наполнить значениями БД,*

*создать аналитические запросы к ней, выгрузить данные*

*в Google таблицу и провести когортный анализ.*

**Этапы создания проекта:**

1. Выбор предметной области и назначения БД.

2. Проектирование БД в dbdesigner.net.

3. Создание БД в Oracle Live SQL.

4. Заполнение БД значениями.

5. Написание запросов для анализа данных.

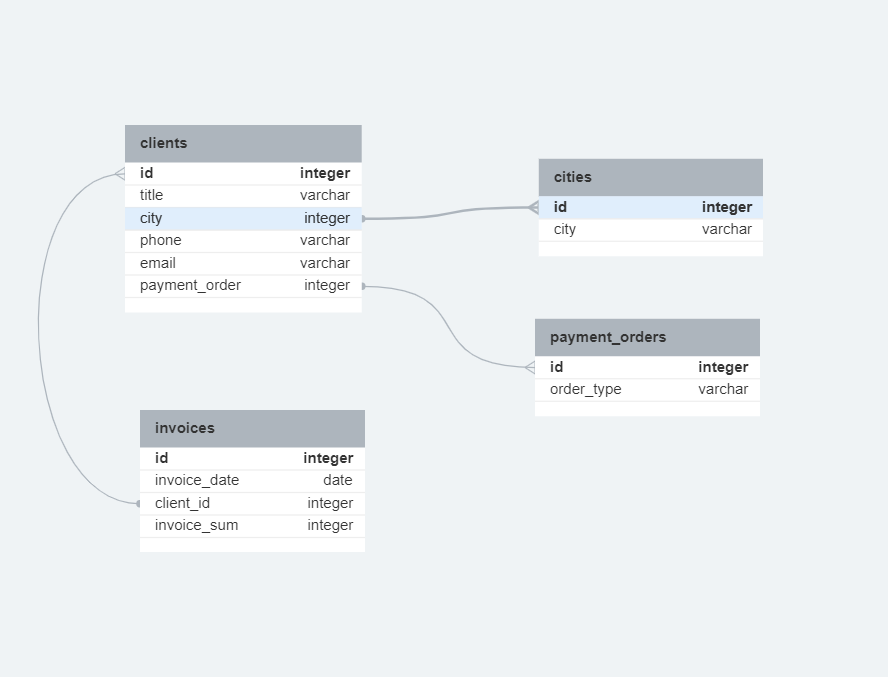
6. Выгрузка данных в Google таблицу и когортный анализ.

**1.** **Выбор предметной области и назначения БД**

В качестве предметной области выбраны адаптированные для целей настоящего проекта и частично обезличенные данные об оптовых продажах сувенирной продукции в Беларуси за 2018 – 2019 год (в период ведения автором проекта индивидуальной предпринимательской деятельности).

Назначение БД – хранение и систематизация данных об отгрузках товара оптовым покупателям.

**2.** **Проектирование БД в dbdesigner.net**

****

<https://erd.dbdesigner.net/designer/schema/1694939337-wholesale-sale-of-souvenirs>

**3.** **Создание БД в Oracle Live SQ**

Из dbdesigner.net был выгружен файл кодом wholesale-sale-of-souvenirs.sql, для удобства дальнейшего использования в коде удалены все кавычки:  
CREATE TABLE payment\_orders (

    id INT NOT NULL,

    order\_type VARCHAR2(70) NOT NULL,

    constraint PAYMENT\_ORDERS\_PK PRIMARY KEY (id));

CREATE sequence PAYMENT\_ORDERS\_ID\_SEQ;

CREATE trigger BI\_PAYMENT\_ORDERS\_ID

  before insert on payment\_orders

  for each row

begin

  select PAYMENT\_ORDERS\_ID\_SEQ.nextval into :NEW.id from dual;

end;

/

CREATE TABLE cities (

    id INT NOT NULL,

    city VARCHAR2(30) NOT NULL,

    constraint CITIES\_PK PRIMARY KEY (id));

CREATE sequence CITIES\_ID\_SEQ;

CREATE trigger BI\_CITIES\_ID

  before insert on cities

  for each row

begin

  select CITIES\_ID\_SEQ.nextval into :NEW.id from dual;

end;

/

CREATE TABLE clients (

    id INT NOT NULL,

    title VARCHAR2(70) NOT NULL,

    city INT NOT NULL,

    phone VARCHAR2(20) NOT NULL,

    email VARCHAR2(30) NOT NULL,

    payment\_order INT NOT NULL,

    constraint CLIENTS\_PK PRIMARY KEY (id));

CREATE sequence CLIENTS\_ID\_SEQ;

CREATE trigger BI\_CLIENTS\_ID

  before insert on clients

  for each row

begin

  select CLIENTS\_ID\_SEQ.nextval into :NEW.id from dual;

end;

/

CREATE TABLE invoices (

    id INT NOT NULL,

    invoice\_date DATE NOT NULL,

    client\_id INT NOT NULL,

    invoice\_sum INT NOT NULL,

    constraint INVOICES\_PK PRIMARY KEY (id));

CREATE sequence INVOICES\_ID\_SEQ;

CREATE trigger BI\_INVOICES\_ID

  before insert on invoices

  for each row

begin

  select INVOICES\_ID\_SEQ.nextval into :NEW.id from dual;

end;

/

ALTER TABLE clients ADD CONSTRAINT clients\_fk0 FOREIGN KEY (city) REFERENCES cities(id);

ALTER TABLE clients ADD CONSTRAINT clients\_fk1 FOREIGN KEY (payment\_order) REFERENCES payment\_orders(id);

ALTER TABLE invoices ADD CONSTRAINT invoices\_fk0 FOREIGN KEY (client\_id) REFERENCES clients(id);

**4.** **Заполнение БД значениями**

Для заполнения БД значениями был подготовлен файл Опт-сувениры.xlsx, для подготовки запросов использованы такие формулы:

=insert into payment\_orders (id, order\_type) values "&D3&";

=insert into cities (id, city) values "&D10&";

=insert into clients (id, title, city, phone, email, payment\_order) values ("&G19&");

=("&A3&", to\_date('"&C3&"', 'yyyy-mm-dd'), "&D3&", "&E3&")

=insert into invoices (id, invoice\_date, client\_id, invoice\_sum) values "&G3&";

и в результате созданы следующие запросы (привожу частично, см. файл inserting.sql):

insert into cities (id, city) values (1, 'Минск');

insert into cities (id, city) values (2, 'Брест');

……

insert into payment\_orders (id, order\_type) values (1, 'Оплата по мере реализации');

insert into payment\_orders (id, order\_type) values (2, 'Срок оплаты 30 дней');

…….  
insert into clients (id, title, city, phone, email, payment\_order) values (1, 'Брестский универмаг', 2, '+375xxxxxxxx1', '1xxx@gmail.com', 1);

insert into clients (id, title, city, phone, email, payment\_order) values (2, 'Брестская союзпечать', 2, '+375xxxxxxxx2', '2xxx@gmail.com', 1);

…….

insert into invoices (id, invoice\_date, client\_id, invoice\_sum) values (1801, to\_date('2018-01-02', 'yyyy-mm-dd'), 12, 1170);

insert into invoices (id, invoice\_date, client\_id, invoice\_sum) values (1802, to\_date('2018-01-17', 'yyyy-mm-dd'), 1, 930);

…….

**5.** **Написание запросов для анализа данных**

Были созданы запросы с целью подготовить данные для последующего когортного анализа (файл queries.sql). Первый запрос создал выборку покупателей, которые за два года сделали не более 5-и заказов или общая сумма продаж которым не превысила 4 000 белорусских рублей:

create view other\_clients as

select cl.title, c.city, p.order\_type, sum(i.invoice\_sum) as INV\_SUM, count(i.id) as INV\_COUNT

from invoices i

INNER JOIN clients cl ON i.client\_id = cl.id

INNER JOIN cities c ON cl.city = c.id

INNER JOIN payment\_orders p ON cl.payment\_order = p.id

group by cl.title, c.city, p.order\_type having count(i.id) < 6 or sum(i.invoice\_sum) < 4001

order by INV\_SUM desc;

select \* from other\_clients;



Вторым запросом было создано представление с выборкой всех необходимых данных отдельно за 2018 и 2019 год по «хорошим покупателям» для последующего когортного анализа:

create view good\_clients as

select cl.title, c.city, p.order\_type,

sum(case when i.invoice\_date between to\_date('2017-12-31', 'yyyy-mm-dd') and to\_date('2019-01-01', 'yyyy-mm-dd')

then i.invoice\_sum else 0 end) as INV\_SUM\_18,

count(case when i.invoice\_date between to\_date('2017-12-31', 'yyyy-mm-dd') and to\_date('2019-01-01', 'yyyy-mm-dd')

then i.id else null end) as INV\_COUNT\_18,

sum(case when i.invoice\_date between to\_date('2018-12-31', 'yyyy-mm-dd') and to\_date('2020-01-01', 'yyyy-mm-dd')

then i.invoice\_sum else 0 end) as INV\_SUM\_19,

count(case when i.invoice\_date between to\_date('2018-12-31', 'yyyy-mm-dd') and to\_date('2020-01-01', 'yyyy-mm-dd')

then i.id else null end) as INV\_COUNT\_19

from invoices i

INNER JOIN clients cl ON i.client\_id = cl.id

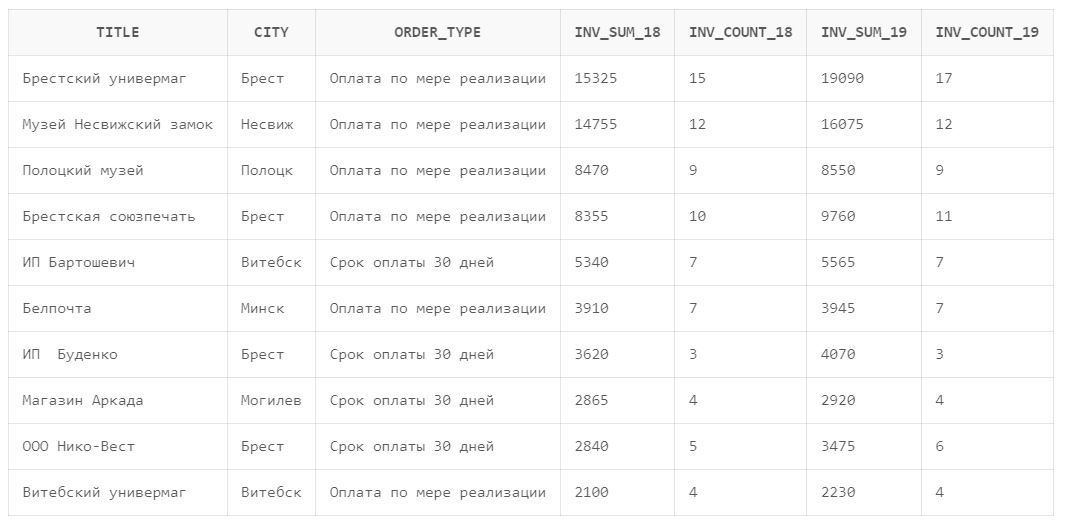
INNER JOIN cities c ON cl.city = c.id

INNER JOIN payment\_orders p ON cl.payment\_order = p.id

group by cl.title, c.city, p.order\_type having count(i.id) > 5 and sum(i.invoice\_sum) > 4000

order by INV\_SUM\_18 desc;

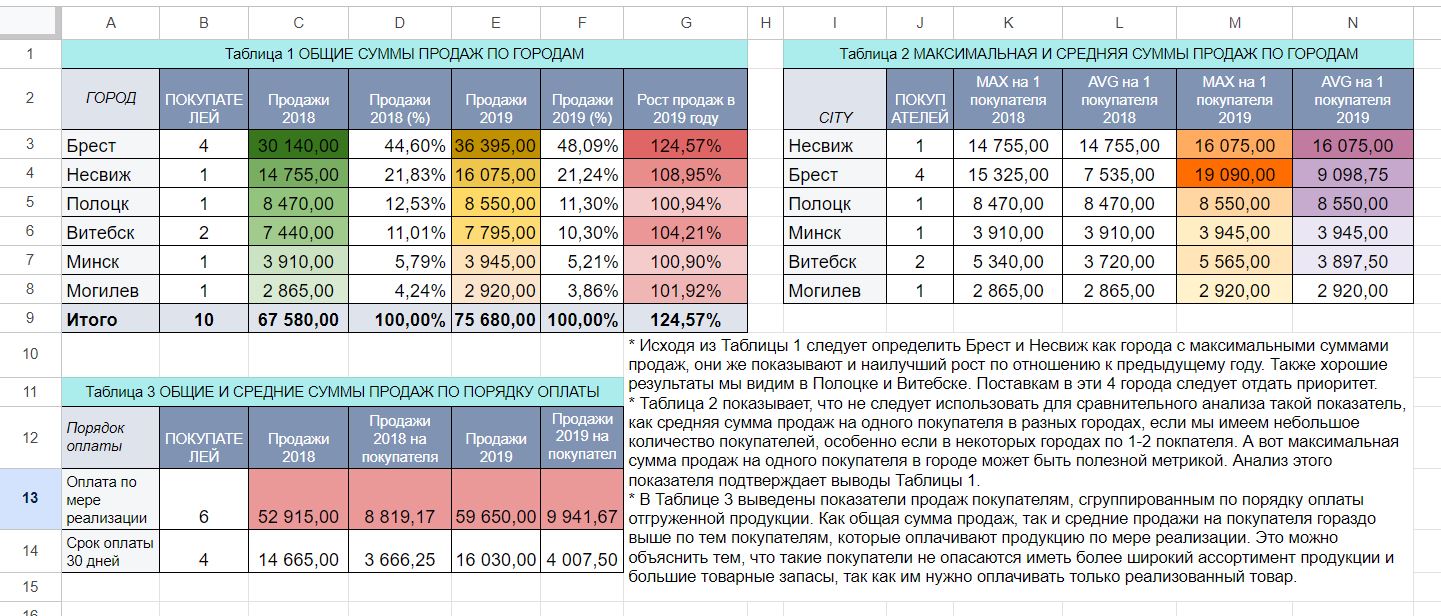
select \* from good\_clients;



**6.** **Выгрузка данных в Google таблицу и когортный анализ**

Представление good\_clients было скачано в формате файла csv, преобразовано в файл table.xsls, данные разбиты по столбцам и скопированы в Google таблицу.

<https://docs.google.com/spreadsheets/d/1pkex_5FJCJxegPoRysJyKpOxllRVuMiIBEHPHr1BraQ/edit#gid=741241786>



\* Исходя из Таблицы 1 следует определить Брест и Несвиж как города с максимальными суммами продаж, они же показывают и наилучший рост по отношению к предыдущему году. Также хорошие результаты мы видим в Полоцке и Витебске. Поставкам в эти 4 города следует отдать приоритет. \* Таблица 2 показывает, что не следует использовать для сравнительного анализа такой показатель, как средняя сумма продаж на одного покупателя в разных городах, если мы имеем небольшое количество покупателей, особенно если в некоторых городах по 1-2 покупателя. А вот максимальная сумма продаж на одного покупателя в городе может быть полезной метрикой. Анализ этого показателя подтверждает выводы Таблицы 1. \* В Таблице 3 выведены показатели продаж покупателям, сгруппированным по порядку оплаты отгруженной продукции. Как общая сумма продаж, так и средние продажи на покупателя гораздо выше по тем покупателям, которые оплачивают продукцию по мере реализации. Это можно объяснить тем, что такие покупатели не опасаются иметь более широкий ассортимент продукции и большие товарные запасы, так как им нужно оплачивать только реализованный товар.