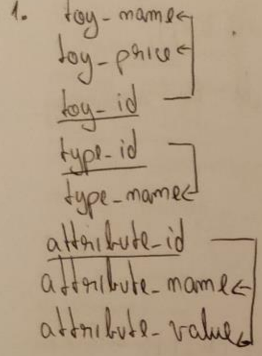
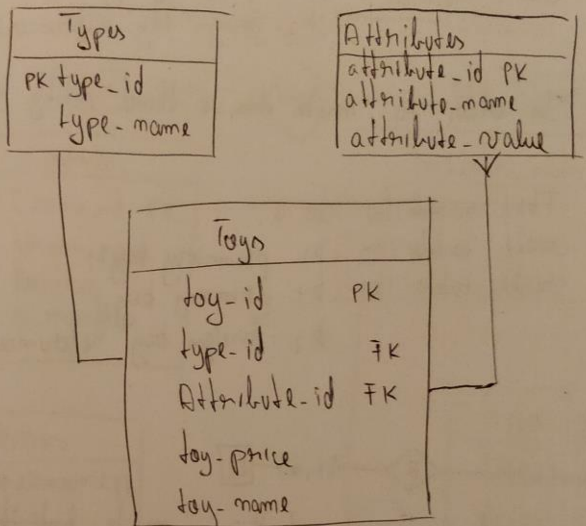
Lab 1

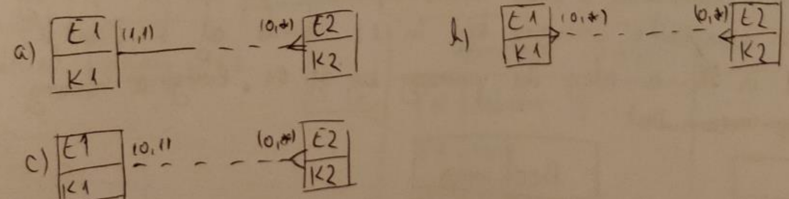
Andrei Nogura
C16733435

1.





2.



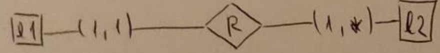
1. I believe this solution is easy to use and maintain because new types and attributes always appear and can be introduced separately and ~~they don't link directl~~ they can be added even if a toy does not have those attributes. ~~& This solution is poy~~

2. a)
```
create table e2 ( K2 primary key);
create table e1 (k1 primary key
                 k2,
                 foreign key (k2) references E2( k2));
```

② b) create table e1 (k1 primary key);
create table e2 (~~k~~ k2 primary key),
k3 foreign key references ~~&~~ e1(k1));

In order to make this, a third entity would be necessary.

c) ~~This is similar to a.~~
create table e2 (k2 primary key);
create table e1 (k1 primary key,
k2 foreign key references e2 (k2));



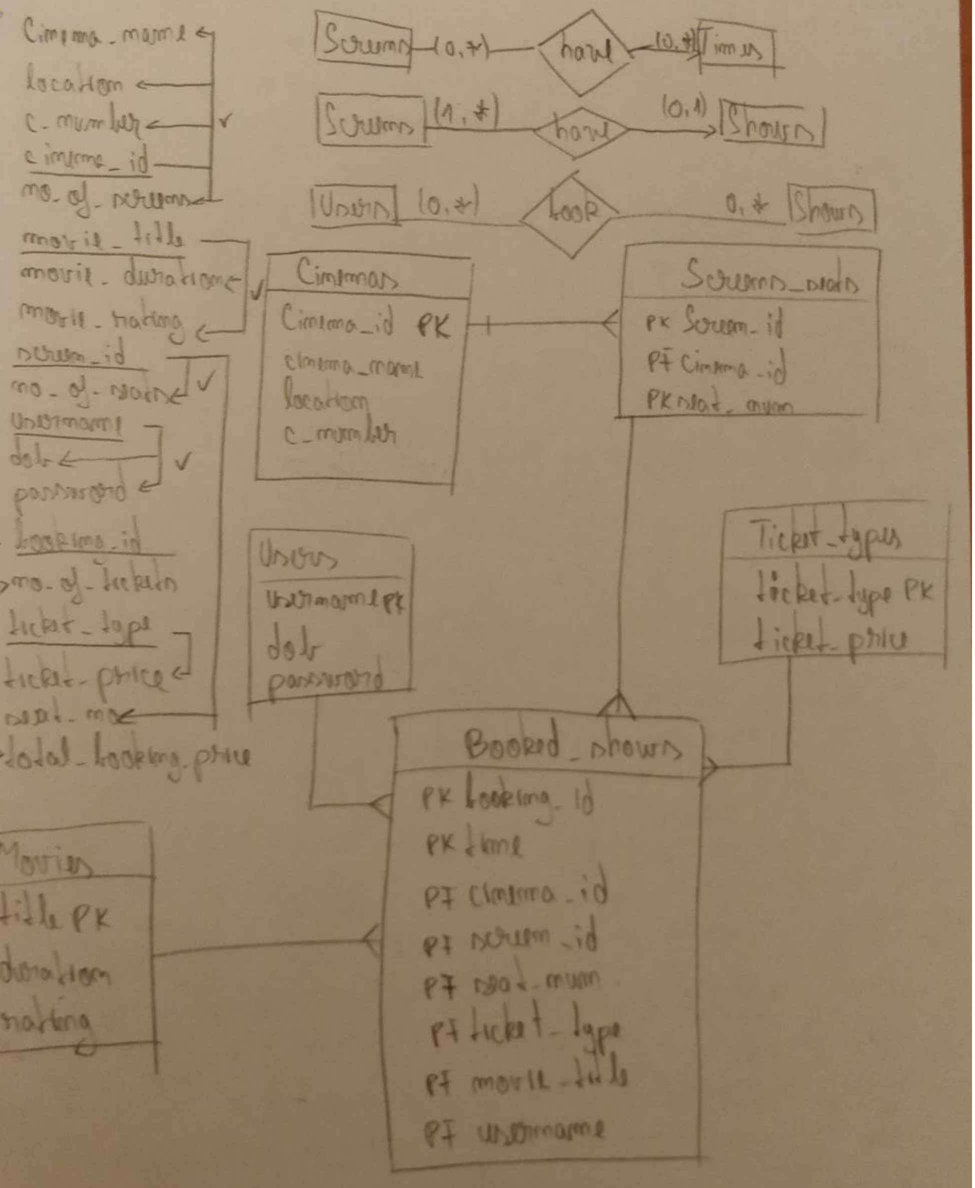
It is possible to implement this in oracle ~~but it is considered a weak entity~~ but it in considered a ~~weak~~ strong relationship. We can achieve this by having a composite primary key in e2 made of k1, and k2 where k1 is the is also the primary key of e1. ~~Being a strong relationship means~~ ~~that~~

3. ~~Cinemas~~
~~Cinema_id PK~~
~~name~~
~~location~~
~~contact_num~~
~~no_of_screens~~

~~Screens~~
~~Screen_id PK~~
~~no_of_seats~~
~~cinema_id PK FK~~

~~Bookings~~
~~booking_id~~
~~Time~~
~~FK screen_id~~
~~FK cinema_id~~
~~FK Movie_id~~
~~seat_no~~
~~FK Username~~
~~total~~
~~no_of_tickets~~

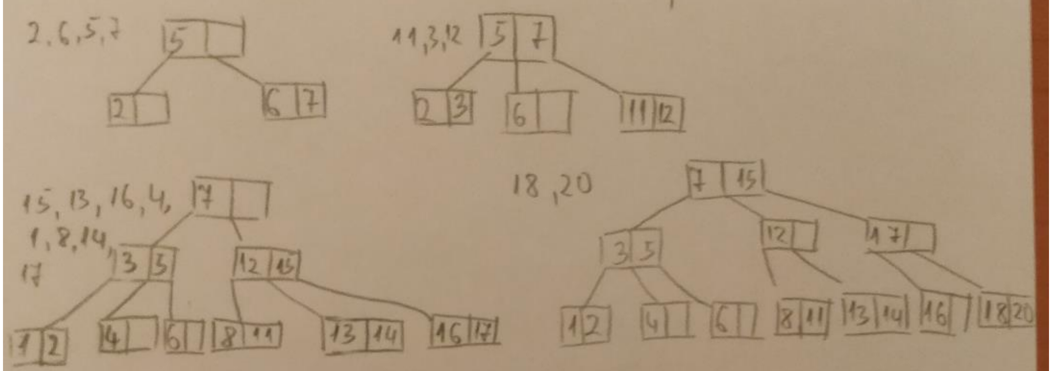
3.


cinema_name
location
c_number
cinema_id
no_of_screens
movie_title
movie_duration
movie_rating
screen_id
no_of_seats
username
dob
password
booking_id
no_of_tickets
ticket_type
ticket_price
seat_no
total_booking_price
✓
✓
✓
✓
Screens
(0,*)
have
(0,*)
Times
Screens
(1,*)
have
(0,1)
Shows
Users
(0,*)
book
0,*
Shows
Cinemas
Cinema_id PK
cinema_name
location
c_number
Screens_seats
PK Screen_id
PF Cinema_id
PK seat_num
Users
username PK
dob
password
Ticket_types
ticket_type PK
ticket_price
Booked_shows
PK booking_id
PK time
PF cinema_id
PF screen_id
PF seat_num
PF ticket_type
PF movie_title
PF username
Movies
title PK
duration
rating

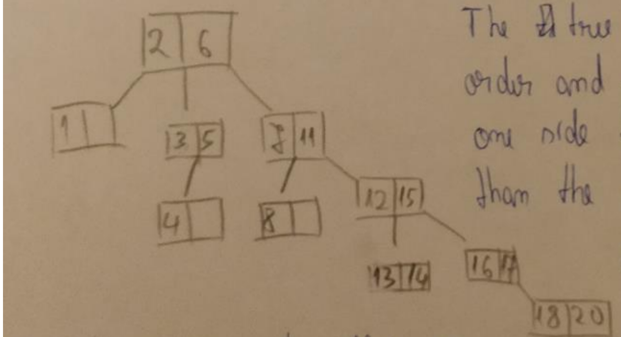
Pag 3

Lab 3

1. 2, 6, 5, 7, 11, 3, 12. | 15, 13, 16, 4, 1, 8, 14, 17 |, 18, 20



2. 2 6 5 7 11 3 12 15 13 16 4 1 8 14 17 18 20



The ~~B~~ tree depends heavily on the input order and ~~tends to lean towards~~ one side tends to be a lot longer than the other. ~~, meaning the~~ This means that the search would be a lot slower

3. Bitmap indexes work well for low cardinality columns. They use bit arrays and answer queries by performing bitwise logical operations. They have significant space and performance.

Although, they are less efficient than b-trees indexes for columns where data is frequently updated. They are mostly employed in data warehousing because ~~its~~ a it is a read-only system specialized for fast query

③ ~~I use the bitmap index on a column wil~~

for

Monthly_sales_report		
Report_no	Month	Sales
1	August	20000
2	September	15000
3	October	21000
4	August	24000

The bitmap is

Month	Bitmap
August	1001
September	0100
October	0010

This is because ~~our~~ August appears in the 1st ~~ad~~ and 4th row, September in the second and October in the third.

For a month bitmap index with 5 million records there are

5.000.000 * 12 = 60.000.000 bits / 8 = 7.500.000 bytes =>

=> 7.15 MB

④ 9, 10, 19 are missing 17 numbers

Sst 2, 6, 5, 7, 11, 3, 12, 15, 13, 16, 4, 1, 8, 14, 17, 18, 20

Btree 3, 3, 2, 1, 3, 2, 2, 1, 3, 3, 3, 3 3, 3, 2, 3, 3

Stree 1 1 2 2 2 2 3 3 4 4 3 2 3 4 4 5 5

Nodes to reach element

~~Avg= 2,5 modes~~

~~Avg= 2,9 modes~~

9 10 19

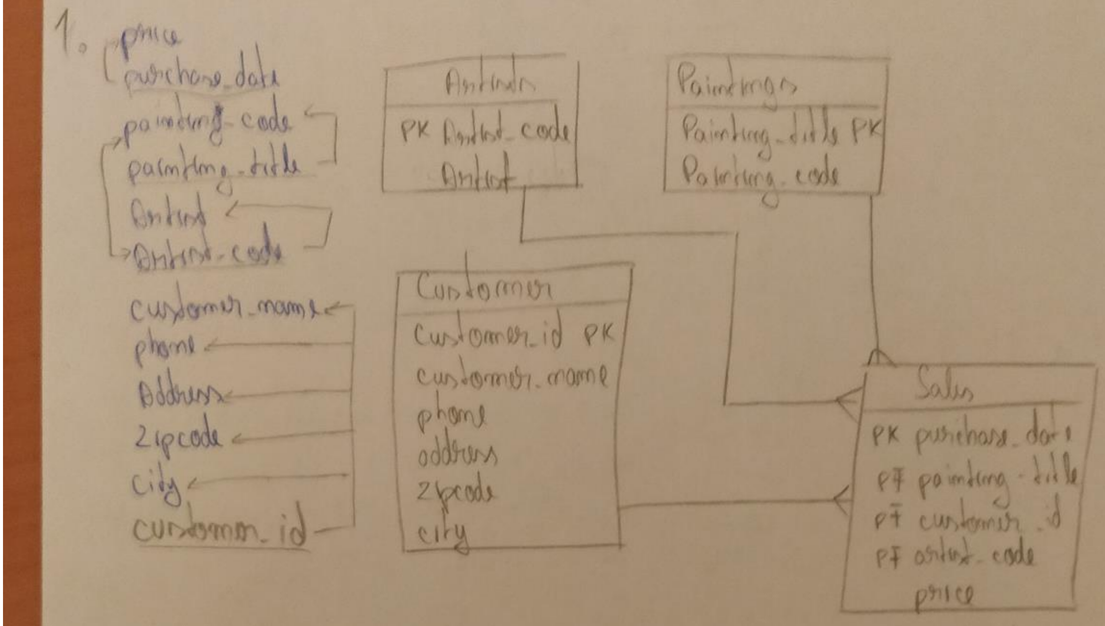
3 3 3 Avg= ~~3.05~~ 2,6

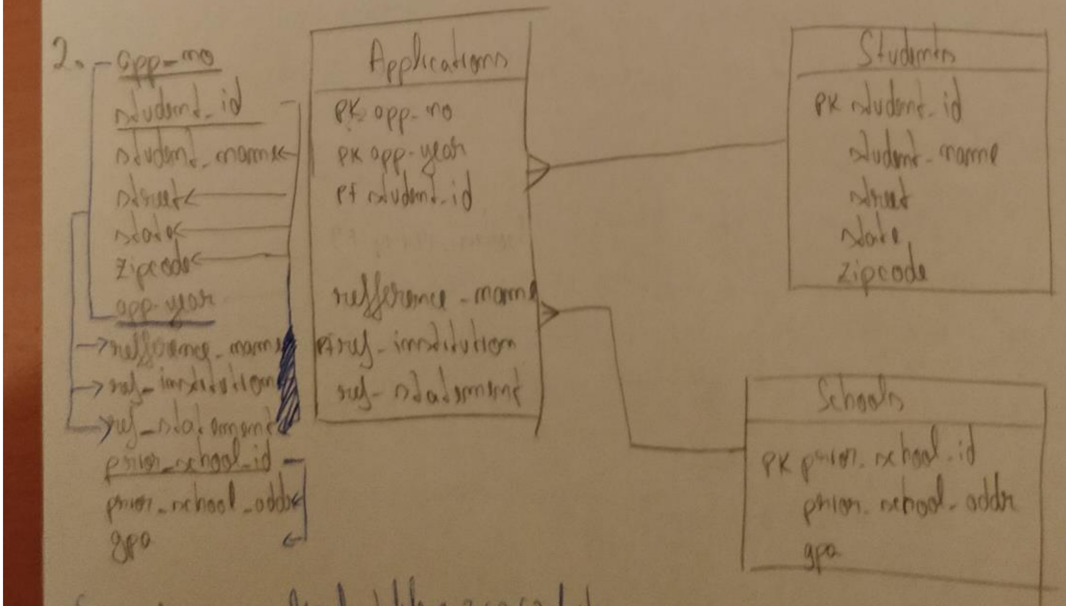
3 3 5 Avg= ~~3.58~~ 3.06

The gain in performance is 17%

3,58 - 3.05 = 0.53 0.53 / 3.05 = 0.17

Lab 2



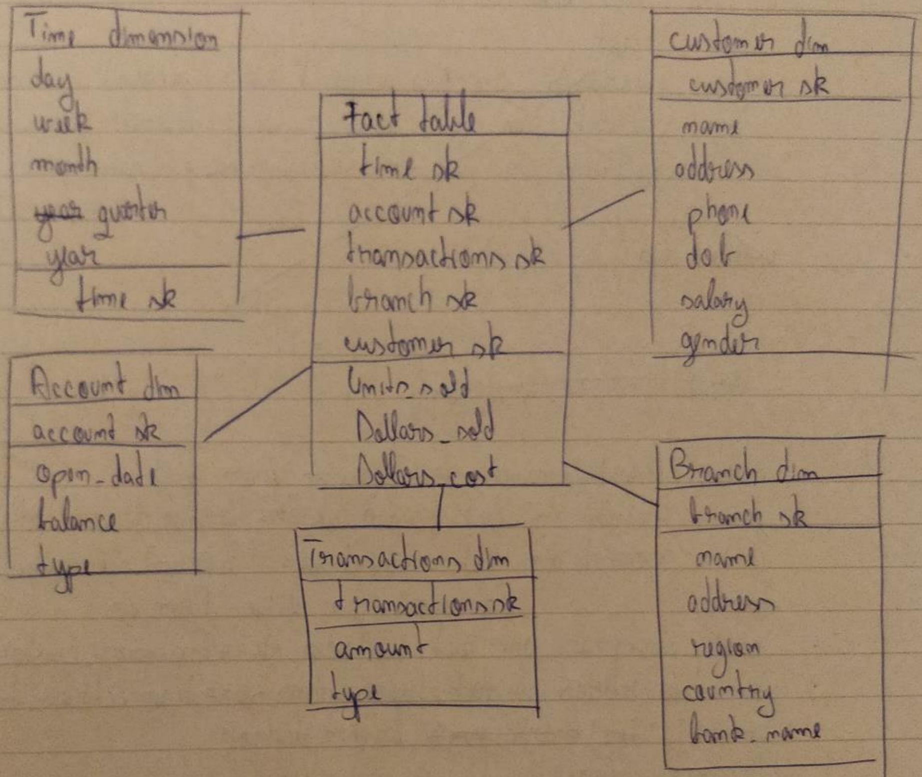


Size of unnormalized table: 36252 bytes
It takes 1007 bytes per row for the unnormalized data and it takes 692 if we put together all the rows after they're normalized so we can say there is a 68% gain in storage efficiency.

Lab 5

1.

a)



b)

```
select sum(amount), fact.time_sk, fact.account_sk, fact.transaction_sk
fact.branch_sk from fact
join account on fact.account_sk == account.account_sk
join time on fact.time_sk == time.time_sk
join transactions on fact.transactions_sk == transactions.transactions_sk
join branch on fact.branch_sk == branch.branch_sk
where account.type == 'Student acc' and
        time.year == 2009
group by( transactions_sk, time_sk, branch_sk, account_sk);
```

c)

```
~~select  sum( amount), fact.account_sk, fact.transactions_sk~~
~~        fact. branch_sk~~
~~from    fact~~
~~join    accounts  on fact.account_sk == account.account_sk~~
~~join    branch  on fact. branch_sk == branch. branch_sk~~
~~join transactions on fact.transactions_sk == transactions.~~
~~                                              transactions_sk~~
~~where  an~~
```

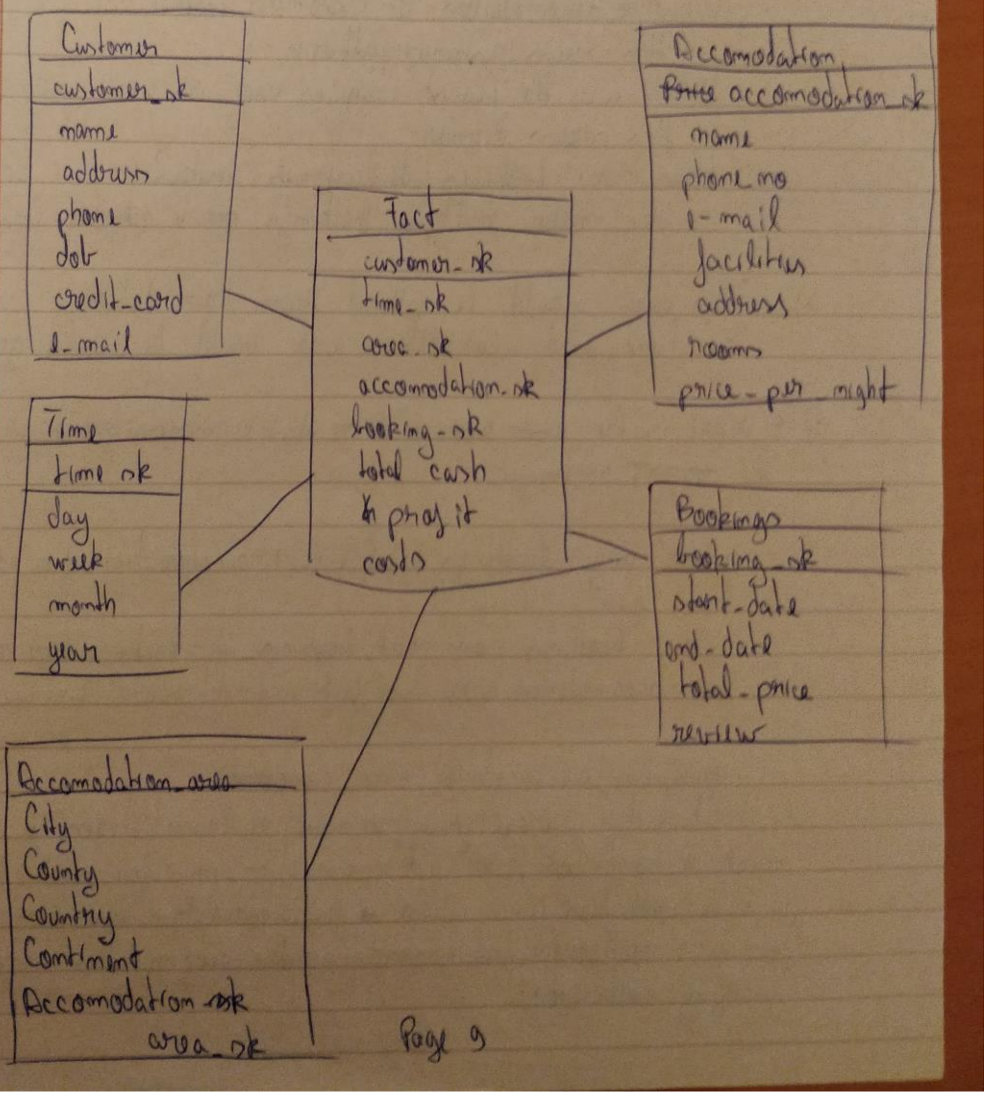
~~What kind of analisys~~

```
select  branch.name , branch.region from fact
join    branch on fact. branch_sk == branch. branch_sk
join    account on fact. account_sk == account. account_sk and
~~join~~                   account.type like 'Premium'
join  customer on fact. customer_sk == customer. customer_sk
join transactions on fact. transactions_sk == transactions. transactions_sk
groupby ( ~~Reg~~ ~~branch. region~~ branch. region)
having  count( customer_sk) >1000 and
        sum( transactions. amount) > 10000;
```

Lab 5

2. c) What reports does ~~[illegible]~~ booking.com need

- ✓ Customers, ~~Businesses~~
- ✓ Time
- ✓ Accomodations
- ✓ Bookings
- ✓ ~~Area~~ Accomodation areas



a) Booking.com is a travel e-commerce website with a mission to make it easier for everyone to travel. It takes the friction out of travel and connects the travelers with the largest selection of incredible places to stay.

~~This model answers these questions and of~~

This model answers but is not limited to these questions only:

What ~~ve~~ accomodations do customers book?

- this shows customer patterns

What are the busiest regions and on what periods?

- this shows trends

How many bookings do customers make?

Do they make multiple bookings on a regular basis?

b) This data would be gathered from their database

The time and accomodation area would be auto-generated

d) • What is the ~~aver~~ average £ ~~so~~ that customers spend on accomodation by ~~region,~~ continent.

```
select avg(bookings.total_price), accomodation_area.continent.
from fact
join   bookings on fact.bookings_sk == bookings.bookings_sk
join accomodation_area on fact.accomodation_area_sk == accomodation_area.
                                                      accomodation_area_sk

groupby(accomodation_area.continent);
```

• What is the average price per night of accomodations by continent.

```
select a.continent, avg(~~£~~accomodation.price_per_night) from fact
join accomodation_area a on ~~•~~ fact.accomodation_area_sk == a. -11-
join accomodation on fact.accomodation_sk == accomodation.accomodation_sk
groupby(continent);
```