CASE SESSION #3

By Iñigo Sanz

TASK B:

Source data:
In the source data, we find the following information:

Category: id, name
Member: id, active, yearActive, gender, want_spam, balance, undo_count
OldPrice: id, product_id, price, changed_on
Payment: id, member_id, timestamp, amount
ProductCategories: id, product_id, category_id
ProductRooms: id, product_id, room_id
Product: id, name, price, active, deactivate_date, quantity, alcohol_quantity_ml, start_date
Room: id, name, description
Sale: id, member_id, product_id, room_id, timestamp, price

We can find information in all of these tables except in *ProductRooms*, where no rows are available.

Business processes:
The business processes we can model are sales, such as the total amount of money gained or the number of products sold, as well as member management including the payments they have made and how regularly they deposit money. Moreover, we can model the products themselves, regarding which ones are the most sold or how their price have fluctuated.

The business process that we have decided to model for this work is the sales of the F-Klub.

Granularity:
The granularity is Sales by Product by Room by Member by Time. The granularity has to be as finer as possible since we want to keep a detailed view of each day's sales.

Are these choices reasonable for a paying customer?
With the granularity that we chose we can know exactly which products, as well as where and when a given member has made a purchase. Not only is it useful from a business analysis outlook, but also for the client since they can check the log of purchases they have made, or compare how much they paid this month compared to previous ones.

Examples of questions:
- What has a member bought in the last week/month
- How much has a member spent in the last week/month
- How many products of a given category were purchased by a member
- In which rooms are most products usually purchased
- How much have the products' prices fluctuated in the last period of time

- How much do members usually deposit
- Which periods of time are most profitable for the F-Klub regarding their sales

All these questions can also be extended to longer periods of time, as well as a group of members instead of a single member etc.

TASK C:

Dimensions:
The dimensions are Product, Room, Member and Time.

The Time dimension has to be very specific to know when a given sale has been done which is why we specify it all the way to the day the sale has been done. Moreover, we take into account the time period of the day to specify if the sale was done in the morning, in the afternoon or in the evening. This way we can know more concretely when in the day sales usually happen. We also added descriptive attributes to know more specifically whether the sale was done during an exam or holiday period, as well as if there was an event in the faculty among other factors. Finally, this dimension allows us to do a roll-up to get a summary of the sales or drill-down for more specificity.

The Product dimension has an additional level that indicates the category of the product. With it we can know which categories the members usually purchase.

The Room dimension allows us to know in which place the product was bought. It doesn't have any attributes or levels since it cannot be correlated to any other dimension (i.e a product can be bought in more than one room) and in the F-Klub sales data, there doesn't exist any more information regarding the room.

The Member dimension doesn't have any levels, however, it keeps a couple of attributes such as the member's gender and the subscription year. This can describe whether old members purchase less than new members or the other way around. We could also keep a loyalty system to reward the oldest members.

The hierarchies for the dimensions are the following:

- Second → Minute → Hour → DayTime → Day → Month → Quarter → Year → T
- Product → Product categories → T
- Room → T
- Member → T

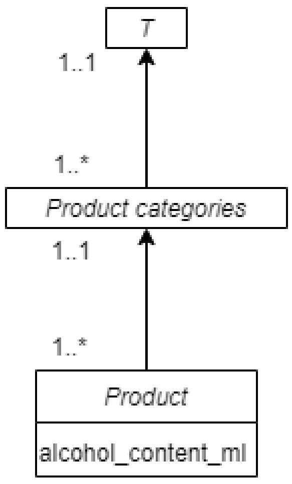
And the dimension attributes are:
Time: *typeOfMonth* (whether it's a holiday, exam period), *event* (a boolean that specifies if there's been a faculty event on a given day).
Product: *alcohol_content_ml* (describes the content of alcohol in a product)
Member: *gender, subscription_year*(the year the member subscribed in)

Measures:
The measures for this business analysis process are the price, and the items sold, and from these, we derive the gross profit and gross margin if the F-Klub supplied us with the information on how much each product cost.
The prices and the items sold are both additive.
The member count is non-additive, because adding the member count through a time period doesn't make sense, nor does adding the member count across the other dimensions.
If we consider the gross profit and gross margin, the former would be additive and the latter non-additive.

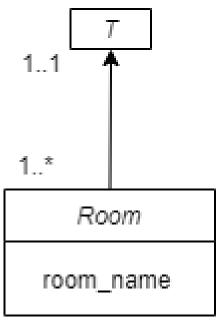
Slow Changing Dimensions:
For this proof of concept, the SCDs that we could have are of type 2 for the products, since their price are fluctuating and it'd be accurate to keep the old prices in case there's some ambiguity with the members' previous purchases.

Dimension designs:

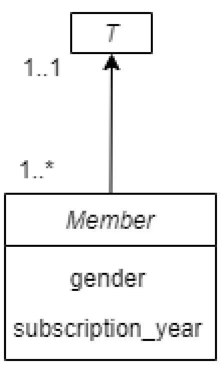
Product:



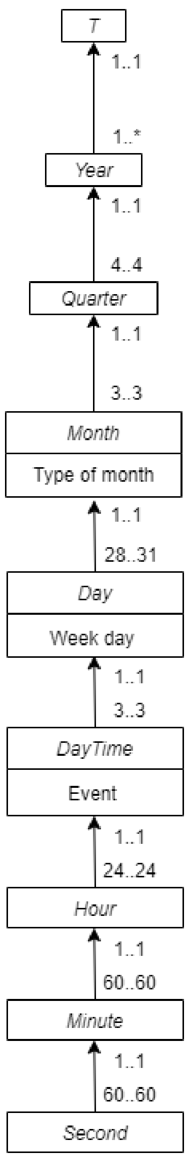
Room:



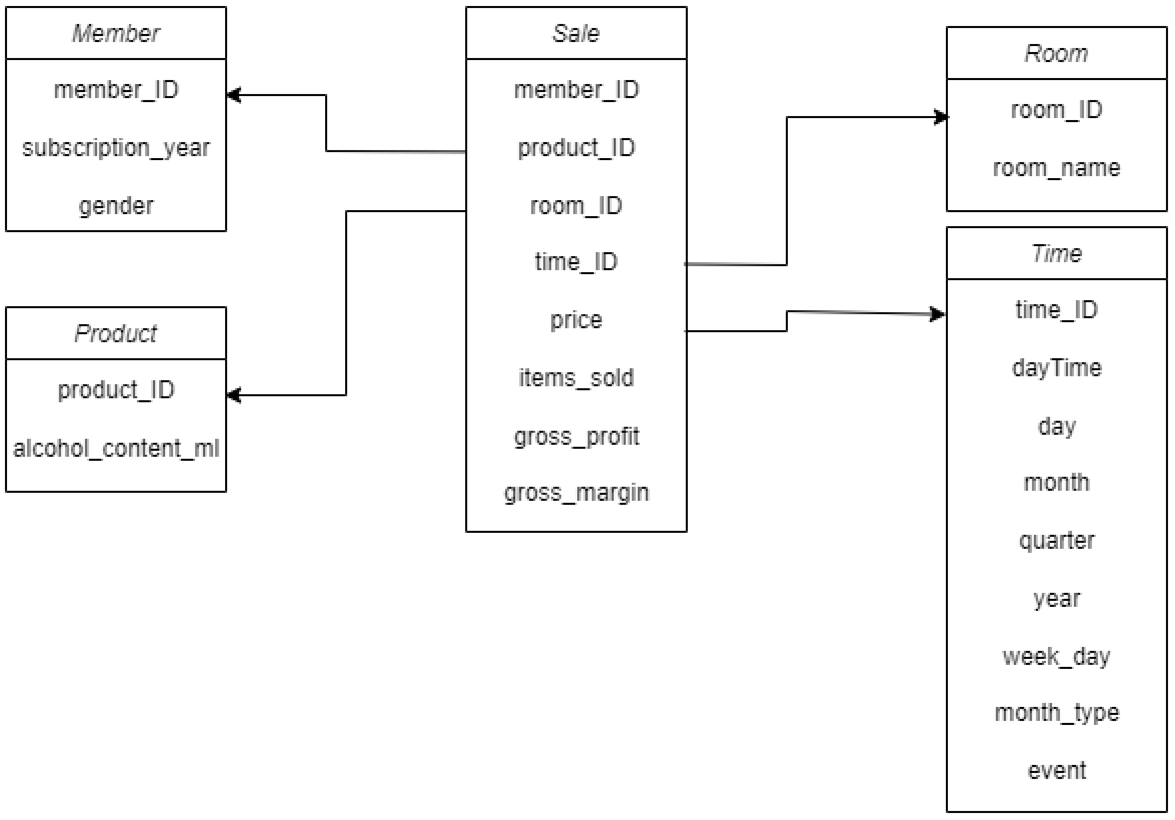
Member:


T
1..1
1..*
Member
gender
subscription_year

Time:


T
1..1
1..*
Year
1..1
4..4
Quarter
1..1
3..3
Month
Type of month
1..1
28..31
Day
Week day
1..1
3..3
DayTime
Event
1..1
24..24
Hour
1..1
60..60
Minute
1..1
60..60
Second

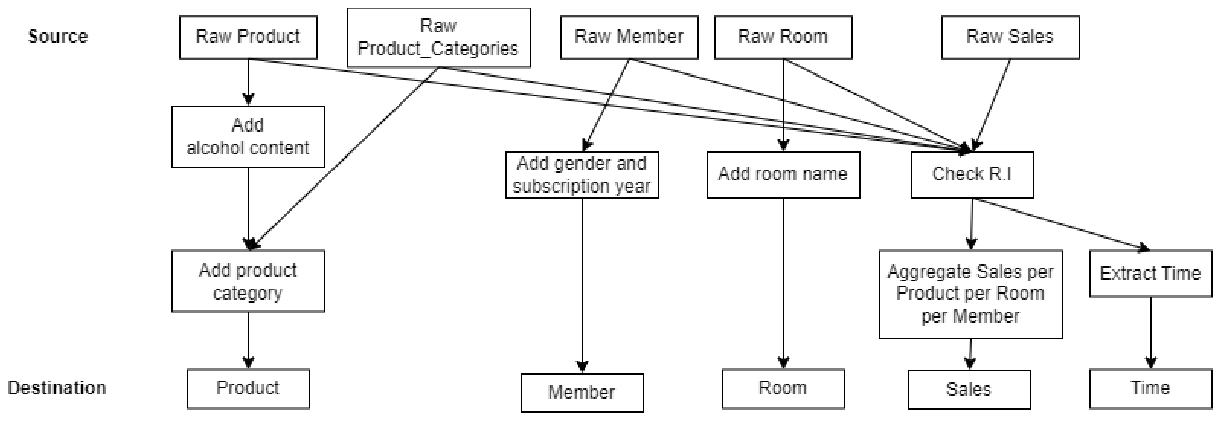
Relational schema:



TASK D:

In order to create an accurate ETL flow, we follow Kimball's ETL Construction Process:

Step 1: Create a high-level diagram of the source-destination flow



Step 2: Choose an ETL tool

The ETL that we chose is pygrametl.

Step 3: Develop default strategies

The data gathered from our source systems comes from the F-Klub. The extracted data should be archived for as long as possible in order to gain insight of the business over the years. Data quality, if possible, should be regulated by a data steward. If the data were to be corrupted, the data steward will try to manually fix it, and in case it's not possible, the data should be discarded. Changes are handled using SCDs of type 2. etc.

The rest of the steps are reflected in the ETL flow.

Regarding the ETL flow:

Firstly, we create connections to the database holding all of the fklub data as well as the destination database for the ETL flow:

```
# Opening of connections and creation of a ConnectionWrapper
sale_conn = psycopg2.connect(host = "localhost", dbname="stregsystem", user="dwuser", password="12345")

dw_conn = psycopg2.connect(host = "localhost", dbname="fklubdw", user="dwuser", password="12345")
dw_conn_wrapper = pygrametl.ConnectionWrapper(connection=dw_conn)
```

With the name mapping, we link each attribute in the SELECT clause to a name of our choosing in the arrays.

```
name_mapping = ['member_id', 'product_id', 'room_id', 'timestamp', 'price', 'count']
name_mapping_member = ['gender', 'subscription_year']
name_mapping_product = ['alcohol_content_ml']
name_mapping_room = ['room_name']

query = "SELECT member_id, product_id, room_id, timestamp, SUM(price), COUNT(*) FROM stregsystem_sale GROUP BY member_id,product_id,room_id,timestamp"
query_member = "SELECT gender, year FROM stregsystem_member"
query_product = "SELECT alcohol_content_ml FROM stregsystem_product"
query_room = "SELECT name FROM stregsystem_room"
```

Then, we create all queries in order to obtain all the necessary data from each of the tables. When creating the SQLSource object, we need to input “SET search_path TO stregsystem” in the initsql parameter so that it obtains the tables from that schema and not from the public one.

```
sale_source = SQLSource(connection=sale_conn, query=query,
        names=name_mapping, initsql='SET search_path TO stregsystem')

sale_source_member = SQLSource(connection=sale_conn, query=query_member,
        names=name_mapping_member, initsql='SET search_path TO stregsystem')

sale_source_product = SQLSource(connection=sale_conn, query=query_product,
        names=name_mapping_product, initsql='SET search_path TO stregsystem')

sale_source_room = SQLSource(connection=sale_conn, query=query_room,
        names=name_mapping_room, initsql='SET search_path TO stregsystem')
```

We create all of the dimensions (CachedDimension to optimize storage) specifying the attributes for each dimension, as well as the fact table with its measures and primary keys. We had to lower the granularity of the time dimension so that each row in the fact table could be have unique primary keys.

```
member_dimension = CachedDimension(
        name='member_',
        key='member_id',
        attributes=['gender', 'subscription_year'])

product_dimension = CachedDimension(
        name='product_',
        key='product_id',
        attributes=['alcohol_content_ml'])

room_dimension = CachedDimension(
        name='room_',
        key='room_id',
        attributes=['room_name'])

time_dimension = CachedDimension(
        name='time_',
        key='time_id',
        attributes=['dayTime', 'second_', 'minute_', 'hour_', 'day_', 'month_', 'quarter', 'year_', 'typeofmonth'])

factTable = FactTable(
    name='sale_',
    measures=['price', 'items_sold'],
    keyrefs=['member_id', 'product_id', 'room_id', 'time_id'])
```

With the split date function, we can obtain from the timestamp measure inside the sale fact table the day, month, year as well as the time the sale was made. From there we can obtain the quarter, as well as the month type, among other things.

```python
def split_date(row):
    timestamp = row['timestamp']
    row['day_'] = timestamp.strftime("%d")
    month = int(timestamp.strftime("%m"))
    row['month_'] = month
    row['year_'] = timestamp.strftime("%Y")
    hour = timestamp.strftime("%H")
    minute = timestamp.strftime("%M")
    second = timestamp.strftime("%S")
    row["hour_"] = hour
    row["minute_"] = minute
    row["second_"] = second
    if (int(hour) < 12):
        row["dayTime"] = "morning"
    else:
        row["dayTime"] = "night"
    if (month <=4):
        row["quarter"] = "Q1"
    elif (month <=8):
        row["quarter"] = "Q2"
    else:
        row["quarter"] = "Q3"
    if (month == 1 or month == 12 or month == 6 or month == 5):
        row["typeofmonth"] = "exam"
    else:
        row["typeofmonth"] = "free"
```

Each row is iterated through in order to fill the remaining information into the dimensions:

```python
for row in sale_source:
    split_date(row)
    row["time_id"] = time_dimension.ensure(row)
    row["items_sold"] = row["count"]
    factTable.insert(row)

for row in sale_source_member:
    row['member_id'] = member_dimension.ensure(row)

for row in sale_source_product:
    row['product_id'] = product_dimension.ensure(row)

for row in sale_source_room:
    row['room_id'] = room_dimension.ensure(row)
```

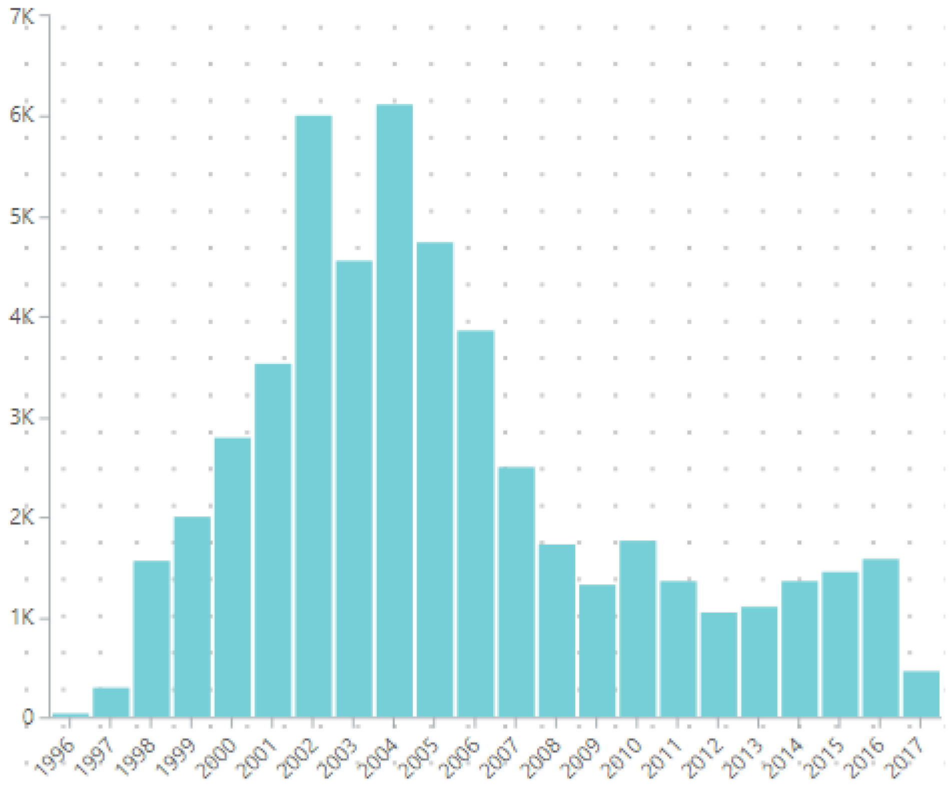
Finally, we close the connections:

```
# The data warehouse connection is then ordered to commit and close
dw_conn_wrapper.commit()
dw_conn_wrapper.close()

# Finally, the connection to the sales database is closed
sale_conn.close()
```

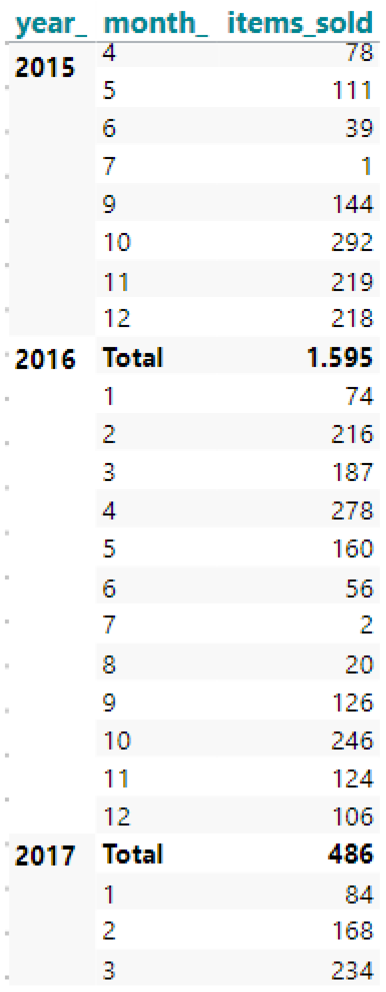
TASK E & F

Graph showing the amount of items sold over the years:



As seen in this graph, we can see that there's been an evolution in the number of items sold from 1996 to 2004, which has decreased until the year 2012 and slowly increased back over the years.
The reason 2017 has so few sales is due to the fact that the sales of 2017 weren't recorded all the way to December. (ends in March as seen in the next figure)

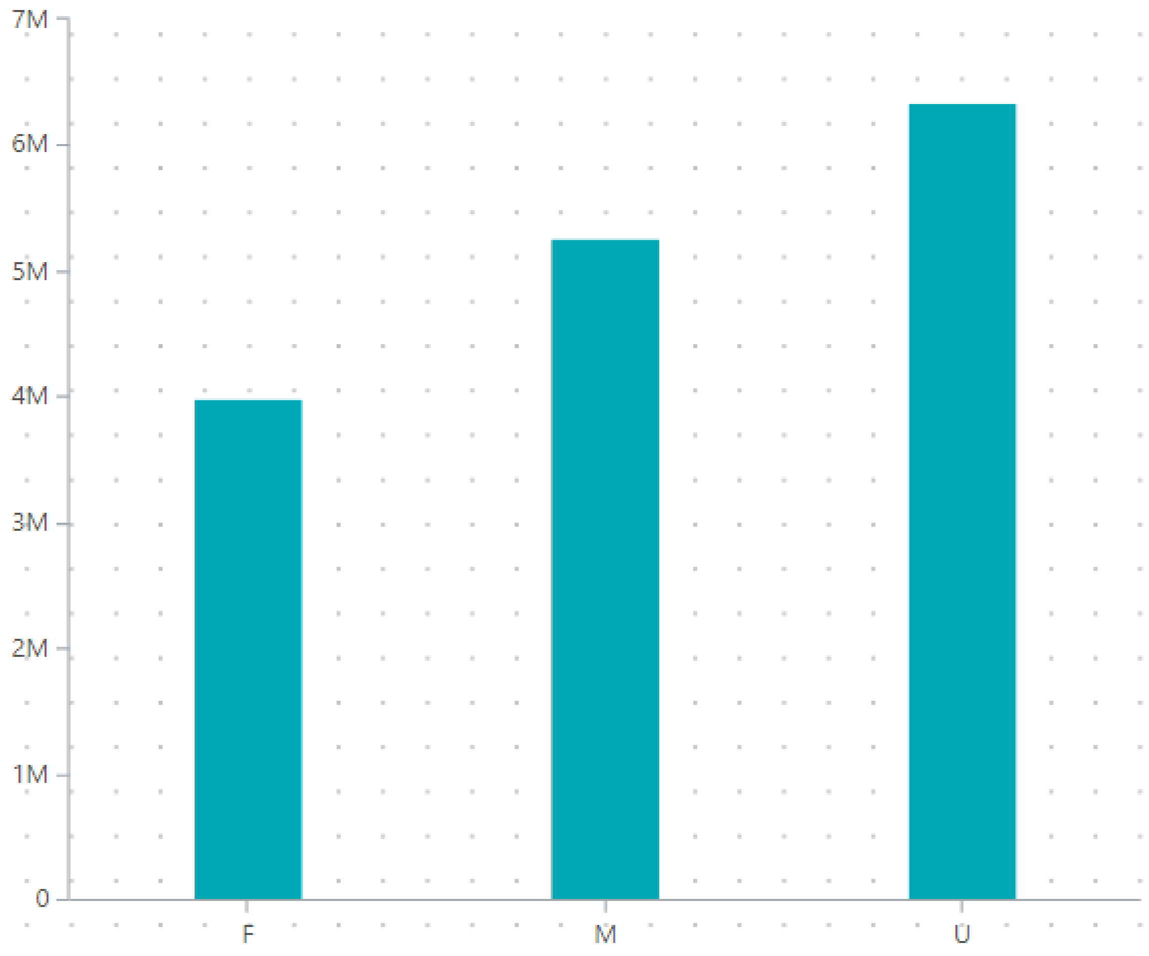
Table with the amount of items sold per month and year:



year_	month_	items_sold
2015	4	78
	5	111
	6	39
	7	1
	9	144
	10	292
	11	219
	12	218
2016	**Total**	**1.595**
	1	74
	2	216
	3	187
	4	278
	5	160
	6	56
	7	2
	8	20
	9	126
	10	246
	11	124
	12	106
2017	**Total**	**486**
	1	84
	2	168
	3	234

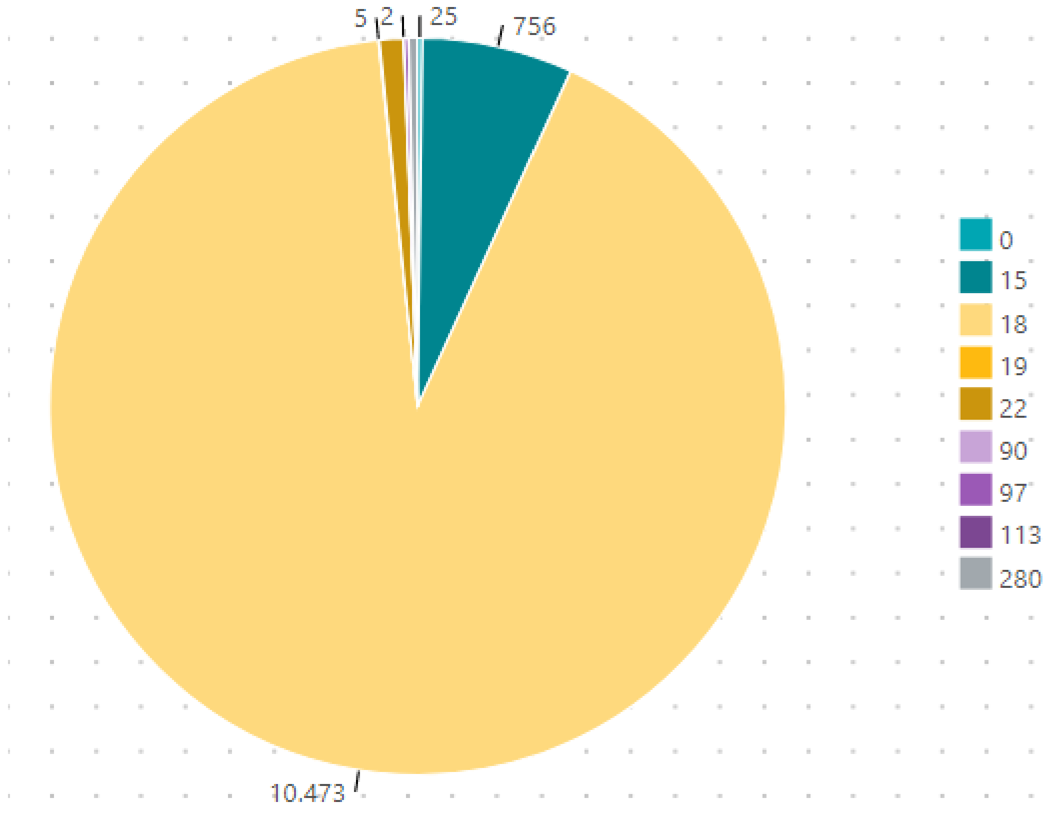
We can get a detailed view of the amount of items sold through the years and months. For example, thanks to this table we know that during the months of July barely any items were sold, due to holidays.

Amount of money spent by male vs female people:



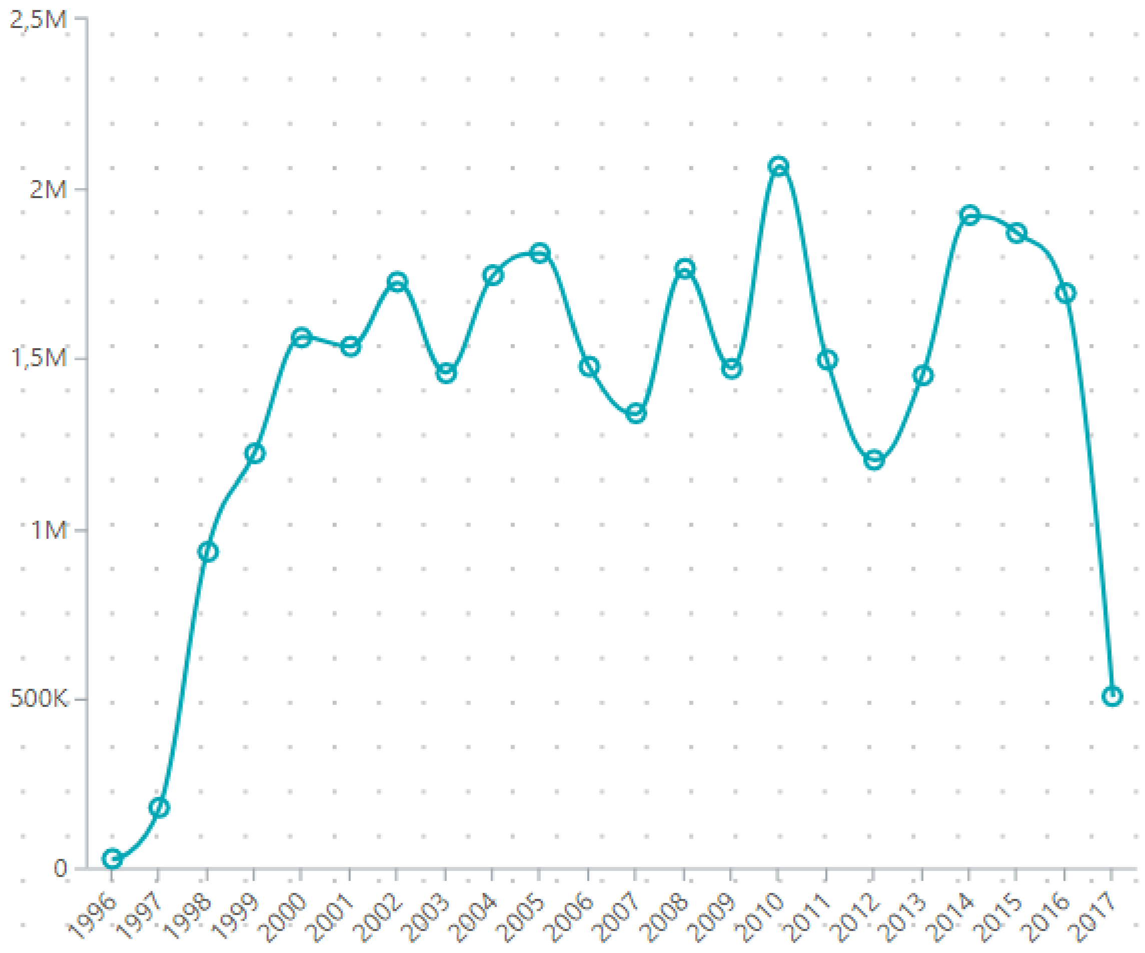
We can know more about our clients, for example whether men spend more than women in the F-Klub. Sadly, most of the clients haven’t specified their gender, therefore the information is somewhat vague.

Amount of items sold by alcohol content (in ml):



This pie chart shows the amount of items sold depending on their alcohol content. As we can see, what most people buy from the F-Klub are alcoholic beverages.

Sales of FKlub over the years:



Finally, this was the goal of this business analysis of the F-Klub. We can observe the amount of money obtained from selling items over the years. It rose up since 1996 and has been steady ever since.