WOLT BI ANALYST ASSIGNEMENT

In [1]:
```
import pymysql
connection = pymysql.connect(user='root', password='', host='localhost')
cursor = connection.cursor()
```

In [2]:
```
DATABASE = 'CREATE DATABASE woltdb'
cursor.execute(DATABASE)
```

In [3]:
```
import pandas as pd
```

In [4]:
```
item_df=pd.read_csv('item_data.csv')
item_df.head(10)
```

Out[4]:

	Unnamed: 0	VENUE_ID	TIMESTAMP	BRAND	MANUFACTURER	COST_PER_UNIT	COST_PER_UNIT_EUR	CURRENCY	APPLICABLE_TAX_PERC	PRODU
0	0	389290dde1	2022-02-01 00:00:00	NaN	Tern1	3.7200	3.720000	EUR	0.200	1e49
1	1	43448c7715	NaN	NaN	NaN	24.8000	1.040115	CZK	0.150	
2	2	6adc2fbb91	NaN	Grøn Balance	VALSEMØLLEN A/S PRIVATE LABEL	6.7400	0.906093	DKK	0.250	740e
3	3	9900b487b1	2022-09-13 00:00:00	ACTIVIA	UAB Eugesta, Kibirkščies g. 8. Vilnius	1.5600	1.560000	EUR	0.210	e0c1
4	4	80ec4c53da	2022-01-01 00:00:00	ACTIVIA	NaN	466.0000	1.128937	HUF	0.180	edd0
5	5	de707394bc	2021-12-01 00:00:00	Vitamineral	Y	0.7400	0.740000	EUR	0.200	

inserting data frame ITEM into the tamporary table ITEM_TEMP

In [6]:
```
from sqlalchemy import create_engine
from mysql.connector import errorcode
import mysql.connector

db_data = 'mysql+pymysql://' + 'root' + ':' + '' + '@' + 'localhost' + ':3306/' \
      + 'woltdb' + '?charset=utf8mb4'

engine=create_engine(db_data)

item_df.to_sql('item_temp', engine,if_exists='replace',index=False)
item="SELECT * FROM item_temp;"
cursor.execute(item)
cursor.fetchall()
```

dropping the unnamed column which we dont need

In [7]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
DROP= """ALTER TABLE `item_temp` DROP `Unnamed: 0`;"""
cursor.execute(DROP)
```

Out[7]: 0

In [8]:
```
purchase_df=pd.read_csv('purchase_data_final.csv')
purchase_df.head(10)
```

Out[8]:

	Unnamed: 0	PURCHASE_ID	TIME_RECEIVED	TIME_DELIVERED	CURRENCY	COUNTRY	VENUE_ID
0	0	c766bf63dc	2022-07-14 06:01:34.426	2022-07-14 06:30:11.423	CZK	CZE	691d84b2f9
1	1	97b4d66216	2022-07-14 08:02:15.789	2022-07-14 08:28:26.345	CZK	CZE	691d84b2f9
2	2	9cadf4d3c4	2022-07-14 06:02:36.614	2022-07-14 06:30:02.425	CZK	CZE	691d84b2f9
3	3	66cf34e8d3	2022-07-14 14:04:45.640	2022-07-14 14:35:00.264	CZK	CZE	691d84b2f9
4	4	94fac08438	2022-07-14 19:04:46.876	2022-07-14 19:26:16.940	CZK	CZE	691d84b2f9
5	5	ebbeb05442	2022-07-14 06:05:02.972	2022-07-14 06:37:53.363	CZK	CZE	691d84b2f9
6	6	d0d8f3f58f	2022-07-14 07:06:01.765	2022-07-14 07:23:27.766	CZK	CZE	691d84b2f9
7	7	8b47d8538a	2022-07-14 12:07:45.994	2022-07-14 12:28:48.297	CZK	CZE	691d84b2f9
8	8	247223c02a	2022-07-14 09:08:18.421	2022-07-14 09:37:48.296	CZK	CZE	691d84b2f9
9	9	f601fcfef5	2022-07-14 12:09:36.258	2022-07-14 12:23:11.041	CZK	CZE	691d84b2f9

inserting purchase dataframe into the tamporart table PURCHASE_TEMP

In [9]:
```
purchase_df.to_sql('purchase_temp', engine,if_exists='append',index=False)
purchase="SELECT * FROM purchase_temp;"
cursor.execute(purchase)
cursor.fetchall()
```

```
  'CZK',
  'CZE',
  '691d84b2f9'),
 (1,
  '97b4d66216',
  '2022-07-14 08:02:15.789',
  '2022-07-14 08:28:26.345',
  'CZK',
  'CZE',
  '691d84b2f9'),
 (2,
  '9cadf4d3c4',
  '2022-07-14 06:02:36.614',
  '2022-07-14 06:30:02.425',
  'CZK',
  'CZE',
  '691d84b2f9'),
 (3,
  '66cf34e8d3',
```

dropping the unnamed columns which we dont need

In [10]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
create_table_query= """ALTER TABLE `purchase_temp` DROP `Unnamed: 0`;"""
cursor.execute(create_table_query)
```

Out[10]: 0

In [11]:
```
purchase_item_df=pd.read_csv('purchase_item_data_final.csv')

purchase_item_df.head(10)
```

Out[11]:

	Unnamed: 0	PRODUCT_ID	PURCHASE_ID	COUNT	VENUE_ID	BASEPRICE	VAT_PERCENTAGE
0	0	0e67b01e73	8d729f3e3a	1	5be413ed1f	1.19	20.0
1	1	24f9c620c6	8d729f3e3a	1	5be413ed1f	2.49	20.0
2	2	27e0da88f2	8d729f3e3a	1	5be413ed1f	2.59	20.0
3	3	7ab5d8bbe6	8d729f3e3a	1	5be413ed1f	2.69	20.0
4	19	2b21ae2ec0	dbc88373f6	6	5c47e55304	22.95	25.0
5	20	0ba3c3e7e2	dbc88373f6	2	5c47e55304	34.50	25.0
6	21	ddd557d878	dbc88373f6	2	5c47e55304	19.50	25.0
7	22	f9c4a9933e	dbc88373f6	1	5c47e55304	21.50	25.0
8	24	ec423edaa2	dbc88373f6	8	5c47e55304	20.95	25.0
9	25	8df279ab62	dbc88373f6	1	5c47e55304	35.95	25.0

inserting purchase item dataframe into the temporary table PURCHASE_ITEM_TEMP

In [12]:
```
purchase_item_df.to_sql('purchase_item_temp', engine,if_exists='append',index=False)
purchase_item="SELECT * FROM purchase_item_temp;"
cursor.execute(purchase_item)
cursor.fetchall()
```

dropping the column Unnamed: 0 which we dont need

In [13]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
DROP= """ALTER TABLE `purchase_item_temp` DROP `Unnamed: 0`;"""
cursor.execute(DROP)
```

creating tables ITEM,PURCHASE and PURCHASE_ITEM

In [14]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
create_purchase_query= """CREATE OR REPLACE TABLE PURCHASE (
    PURCHASE_ID varchar(20)  NOT NULL ,
    TIME_DELIVERED datetime  ,
    TIME_RECEIVED datetime  ,
    CURRENCY varchar(5)  ,
    COUNTRY varchar(20)  ,
    VENUE_ID varchar(20)

);"""
cursor.execute(create_purchase_query)
```

Out[14]:
```
0
```

In [15]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
create_item_query= """CREATE OR REPLACE TABLE ITEM (
    VENUE_ID varchar(20)  ,
    AVAILABLE_TIMESTAMP datetime  ,
    BRAND varchar(500)  ,
    MANUFACTURER varchar(500)  ,
    COST_PER_UNIT Float   ,
    COST_PER_UNIT_EUR Float   ,
    CURRENCY varchar(20)  ,
    APPLICABLE_TAX_PERC Float  ,
    PRODUCT_ID varchar(20)  NOT NULL ,
    ITEM_IDENTIFIER varchar(50)  ,
    EXTERNAL_ID varchar(50)


);"""
cursor.execute(create_item_query)
```

Out[15]:
```
0
```

In [16]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
create_purchase_item_query= """CREATE OR REPLACE TABLE PURCHASE_ITEM (
    PRODUCT_ID varchar(20) NOT NULL ,
    PURCHASE_ID varchar(20) NOT NULL ,
    PURCHASE_COUNT int  ,
    VENUE_ID varchar(20)  ,
    BASEPRICE double(12,2) ,
    VAT_PERCENTAGE FLOAT
);
"""
cursor.execute(create_purchase_item_query)
```

importing data from tamporary tables into the new tables

In [32]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
INSERT_INTO_ITEM= """INSERT INTO item( VENUE_ID, AVAILABLE_TIMESTAMP, BRAND, MANUFACTURER, COST_PER_UNIT, COST_PER_UNIT_EUR, CURR
SELECT VENUE_ID, TIMESTAMP, BRAND, MANUFACTURER, COST_PER_UNIT, COST_PER_UNIT_EUR, CURRENCY, APPLICABLE_TAX_PERC, PRODUCT_ID, ITE
FROM item_temp
WHERE NOT EXISTS( SELECT VENUE_ID, AVAILABLE_TIMESTAMP, BRAND, MANUFACTURER, COST_PER_UNIT, COST_PER_UNIT_EUR, CURRENCY, APPLICAE
FROM item
WHERE item_temp.VENUE_ID = item.VENUE_ID
AND item_temp.TIMESTAMP = item.AVAILABLE_TIMESTAMP
AND item_temp.BRAND = item.BRAND AND item_temp.MANUFACTURER = item.MANUFACTURER
AND item_temp.COST_PER_UNIT = item.COST_PER_UNIT
AND item_temp.COST_PER_UNIT_EUR = item.COST_PER_UNIT_EUR
AND item_temp.CURRENCY = item.CURRENCY
AND item_temp.APPLICABLE_TAX_PERC = item.APPLICABLE_TAX_PERC
AND item_temp.PRODUCT_ID = item.PRODUCT_ID
AND item_temp.ITEM_IDENTIFIER = item.ITEM_IDENTIFIER
AND item_temp.EXTERNAL_ID = item.EXTERNAL_ID
)AND PRODUCT_ID IS NOT NULL;"""

cursor.execute(INSERT_INTO_ITEM)
```

In [46]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
INSERT_PURCHASE_ITEM= """INSERT INTO Purchase_item( PRODUCT_ID, PURCHASE_ID, PURCHASE_COUNT, VENUE_ID , BASEPRICE, VAT_PERCENTAGE
SELECT PRODUCT_ID, PURCHASE_ID, COUNT , VENUE_ID , BASEPRICE, VAT_PERCENTAGE
FROM purchase_item_temp
WHERE NOT EXISTS( SELECT PRODUCT_ID, PURCHASE_ID, PURCHASE_COUNT , VENUE_ID , BASEPRICE, VAT_PERCENTAGE
FROM purchase_item
WHERE purchase_item_temp.PRODUCT_ID=purchase_item.product_id
AND purchase_item_temp.PURCHASE_ID=purchase_item.purchase_id
AND purchase_item_temp.COUNT=purchase_item.PURCHASE_COUNT
AND purchase_item_temp.VENUE_ID=purchase_item.VENUE_ID
AND purchase_item_temp.BASEPRICE=purchase_item.BASEPRICE
AND purchase_item_temp.VAT_PERCENTAGE=purchase_item.VAT_PERCENTAGE )
AND PRODUCT_ID IS NOT NULL AND PURCHASE_ID IS NOT NULL;"""
cursor.execute(INSERT_PURCHASE_ITEM)
```

In [45]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
INSERT_PURCHASE= """INSERT INTO purchase( PURCHASE_ID, TIME_DELIVERED, TIME_RECEIVED, CURRENCY, COUNTRY, VENUE_ID )
SELECT PURCHASE_ID, TIME_DELIVERED, TIME_RECEIVED, CURRENCY, COUNTRY, VENUE_ID
FROM purchase_temp
WHERE NOT EXISTS( SELECT PURCHASE_ID, TIME_DELIVERED, TIME_RECEIVED, CURRENCY, COUNTRY, VENUE_ID
FROM purchase
WHERE purchase_temp.PURCHASE_ID=purchase.PURCHASE_ID
AND purchase_temp.TIME_DELIVERED=purchase_temp.TIME_DELIVERED
AND purchase_temp.TIME_RECEIVED=purchase.TIME_RECEIVED
AND purchase_temp.CURRENCY=purchase.CURRENCY
AND purchase_temp.COUNTRY=purchase.COUNTRY
AND purchase_temp.VENUE_ID=purchase.VENUE_ID )AND PURCHASE_ID IS NOT NULL;"""
cursor.execute(INSERT_PURCHASE)
```

```
--QUERIES USED TO COUNT COSTS and COSTS IN EUR

SELECT (COST_PER_UNIT_EUR*(1 + APPLICABLE_TAX_PERC ))as COST_PER_UNIT_EUR_VAT
FROM PURCHASE_ITEM AS PUI
JOIN ITEM AS I
ON PUI.PRODUCT_ID = I.PRODUCT_ID;

SELECT (COST_PER_UNIT*(1 + APPLICABLE_TAX_PERC ))as COST_PER_UNIT_VAT
FROM PURCHASE_ITEM AS PUI
JOIN ITEM AS I
ON PUI.PRODUCT_ID = I.PRODUCT_ID;

--TOTAL COSTS

SELECT SUM(COST_PER_UNIT_VAT) as TOTAL_COSTS FROM item GROUP BY PRODUCT_ID;

SELECT SUM(COST_PER_UNIT_EUR_VAT) as TOTAL_COSTS_EUR FROM item GROUP BY PRODUCT_ID;

--PRODUCT PRICE

SELECT SUM(BASEPRICE) as TOTAL_PRODUCT_PRICE

--PRODUCT PRICES CONVERTED TO EURO

SELECT (COST_PER_UNIT_EUR/COST_PER_UNIT)as CURRENCY_RATE from ITEM

(BASEPRICE)*(COST_PER_UNIT_EUR/COST_PER_UNIT) AS PRODUCT_PRICE_EUR,

SUM((BASEPRICE)*(COST_PER_UNIT_EUR/COST_PER_UNIT)) AS TOTAL_PRODUCT_PRICE_EUR

--PRODUCT QUANTITY

SELECT PRODUCT_ID, SUM(PURCHASE_COUNT) as Product_Quantity FROM purchase_item GROUP BY PRODUCT_ID;
```

```
--CREATING VIEWS FOR TASK 1 PROFITABILITY FOR EACH PURCHASE

CREATE OR REPLACE VIEW PURCHASE_PROFIT AS
SELECT P.PURCHASE_ID AS PURCHASE_ID,
       PUI.PRODUCT_ID as PRODUCT_ID,
       PUI.VENUE_ID as VENUE_ID,
       P.COUNTRY AS COUNTRY,
       PUI.PURCHASE_COUNT AS PURCHASE_QUANTITY,
       (COST_PER_UNIT_EUR*(1 + APPLICABLE_TAX_PERC )) as COST_PER_UNIT_EUR_VAT,
       (BASEPRICE)*(COST_PER_UNIT_EUR/COST_PER_UNIT)AS PRODUCT_PRICE_EUR
FROM PURCHASE_ITEM AS PUI
JOIN ITEM AS I
ON PUI.PRODUCT_ID = I.PRODUCT_ID
JOIN PURCHASE AS P
ON PUI.VENUE_ID = P.VENUE_ID
WHERE COST_PER_UNIT IS NOT NULL
AND COST_PER_UNIT_EUR IS NOT NULL
AND PURCHASE_COUNT IS NOT NULL

---PROFITABILITY FOR EACH PURCHASE

CREATE OR REPLACE VIEW TOTAL_PROFITABILITY_BY_PURCHASE AS
SELECT PURCHASE_ID,
PRODUCT_ID,
VENUE_ID,
COUNTRY,
SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY)AS TOTAL_REVENUE,
(SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))-SUM(COST_PER_UNIT_EUR_VAT) AS TOTAL_PROFIT,
((SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))-SUM(COST_PER_UNIT_EUR_VAT))*100/(SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))AS TOTAL_MARGIN
FROM PURCHASE_PROFIT
GROUP BY PURCHASE_ID
ORDER BY TOTAL_MARGIN
```

In [51]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
VIEW_PURCHASE_PROFIT= """CREATE OR REPLACE VIEW PURCHASE_PROFIT AS
SELECT P.PURCHASE_ID AS PURCHASE_ID,
PUI.PRODUCT_ID as PRODUCT_ID,
PUI.VENUE_ID as VENUE_ID,
P.COUNTRY AS COUNTRY,
P.TIME_DELIVERED AS TIME_DELIVERED,
(COST_PER_UNIT*(1 + APPLICABLE_TAX_PERC )) as COST_PER_UNIT_VAT,
(COST_PER_UNIT_EUR*(1 + APPLICABLE_TAX_PERC )) as COST_PER_UNIT_EUR_VAT,
PURCHASE_COUNT as PURCHASE_QUANTITY,
(BASEPRICE+(BASEPRICE*VAT_PERCENTAGE))*((COST_PER_UNIT_EUR*(1 + APPLICABLE_TAX_PERC ))/(COST_PER_UNIT*(1 + APPLICABLE_TAX_PERC ))
FROM PURCHASE_ITEM AS PUI
JOIN ITEM AS I
ON PUI.PRODUCT_ID = I.PRODUCT_ID
JOIN PURCHASE AS P
ON PUI.VENUE_ID = P.VENUE_ID
WHERE COST_PER_UNIT IS NOT NULL AND COST_PER_UNIT_EUR IS NOT NULL AND PURCHASE_COUNT IS NOT NULL AND BASEPRICE IS NOT NULL
GROUP BY PURCHASE_ID;"""
cursor.execute(VIEW_PURCHASE_PROFIT)
```

In [52]:
```
use_database_query = "USE woltdb"
cursor.execute(use_database_query)
PROFITABILITY_BY_PURCHASE= """CREATE OR REPLACE VIEW TOTAL_PROFITABILITY_BY_PURCHASE AS
SELECT PURCHASE_ID,
PRODUCT_ID,
VENUE_ID,
COUNTRY,
SUM(COST_PER_UNIT_EUR_VAT) AS TOTAL_COSTS,
SUM(PRODUCT_PRICE_EUR) AS TOTAL_PRODUCT_PRICE,
SUM(PURCHASE_QUANTITY) AS TOTAL_PURCHASE_QUANTITY,
(SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))-SUM(COST_PER_UNIT_EUR_VAT) AS TOTAL_PROFIT,
((SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))-SUM(COST_PER_UNIT_EUR_VAT))*100/(SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))A
FROM PURCHASE_PROFIT
GROUP BY PURCHASE_ID
ORDER BY TOTAL_MARGIN;"""
cursor.execute(PROFITABILITY_BY_PURCHASE)
```

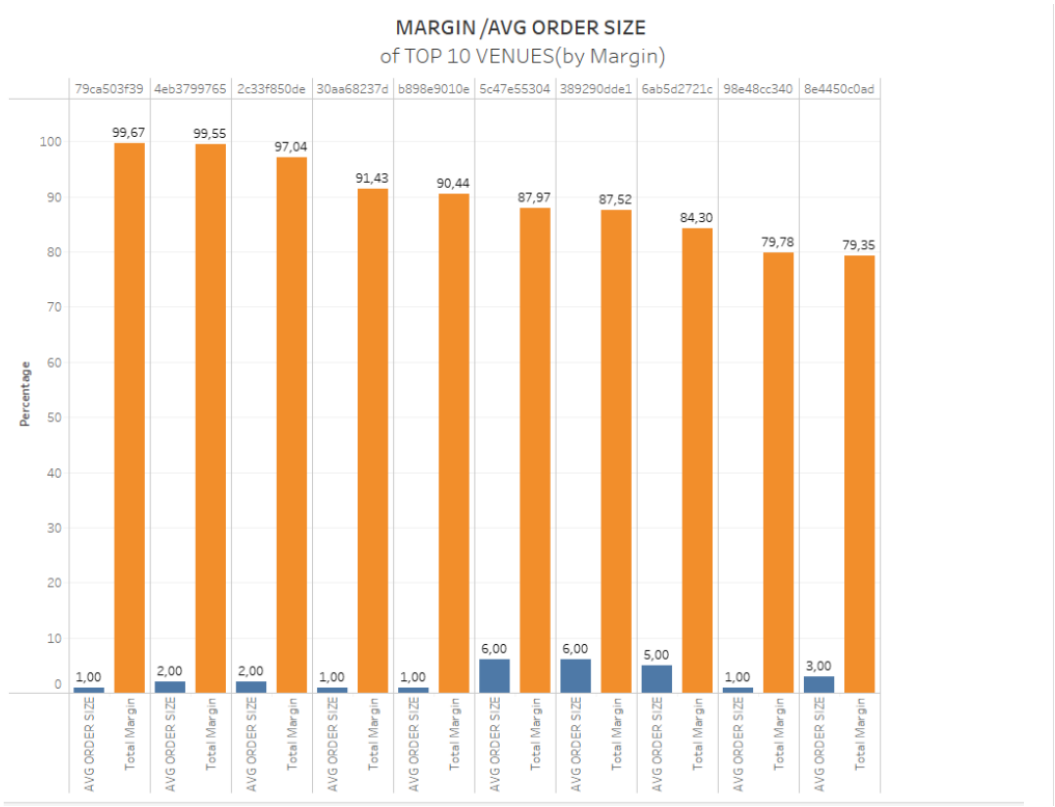
REVENUE=PRODUCT_PRICE*PRODUCT_QUANTITY
TOTAL PROFIT=TOTAL REVENUE-TOTAL_COSTS
TOTAL MARGIN=TOTAL PROFIT/TOTAL REVENUE
AOV (avg order volume)=TOTAL_REVENUE/TOTAL_PRODUCT_QUANTITY IN CERTAIN DATA RANGE

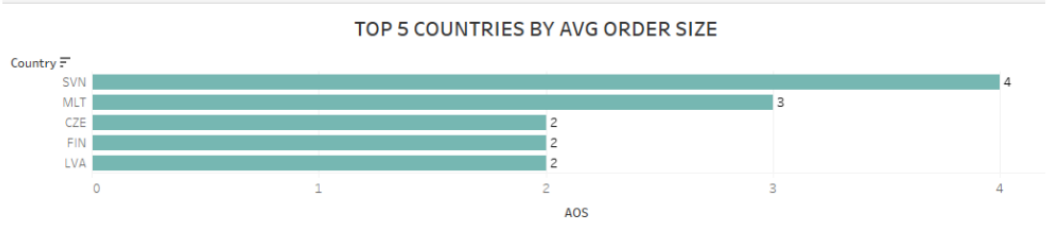
```
--TOP 10 VENUES
CREATE OR REPLACE VIEW PRODUCT_PROFIT_VENUES AS
SELECT PUI.PRODUCT_ID as PRODUCT_ID,
    PUI.VENUE_ID as VENUE_ID,
    P.COUNTRY AS COUNTRY,
    PURCHASE_COUNT as PURCHASE_QUANTITY,
    PUI.PURCHASE_ID as PURCHASE_ID
   (COST_PER_UNIT_EUR*(1 + APPLICABLE_TAX_PERC )) as COST_PER_UNIT_EUR_VAT,
   (BASEPRICE)*(COST_PER_UNIT_EUR/COST_PER_UNIT) AS PRODUCT_PRICE_EUR
FROM PURCHASE_ITEM AS PUI
JOIN ITEM AS I
    ON PUI.PRODUCT_ID = I.PRODUCT_ID
JOIN PURCHASE AS P
    ON PUI.VENUE_ID = P.VENUE_ID
WHERE COST_PER_UNIT_EUR IS NOT NULL
AND PURCHASE_COUNT IS NOT NULL
AND BASEPRICE IS NOT NULL
GROUP BY VENUE_ID


CREATE OR REPLACE VIEW TOTAL_PROFIT_EUR_VENUES AS
SELECT PURCHASE_ID,
       PRODUCT_ID,
       VENUE_ID,
       AVG(PURCHASE_QUANTITY) AS AVG_PURCHASE_SIZE
       SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY) AS TOTAL_PRODUCT_REVENUE,
       (SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))-SUM(COST_PER_UNIT_EUR_VAT) AS TOTAL_PROFIT,
      ((SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))-SUM(COST_PER_UNIT_EUR_VAT))*100/(SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY))AS TOTAL_MARGIN
FROM PRODUCT_PROFIT_VENUES
GRoup by VENUE_ID
ORDER BY TOTAL_MARGIN DESC


select VENUE_ID, AVG_PURCHASE_SIZE, TOTAL_MARGIN
from TOTAL_PROFIT_EUR_VENUES
group by VENUE_ID
order by TOTAL_MARGIN desc LIMIT 10
```

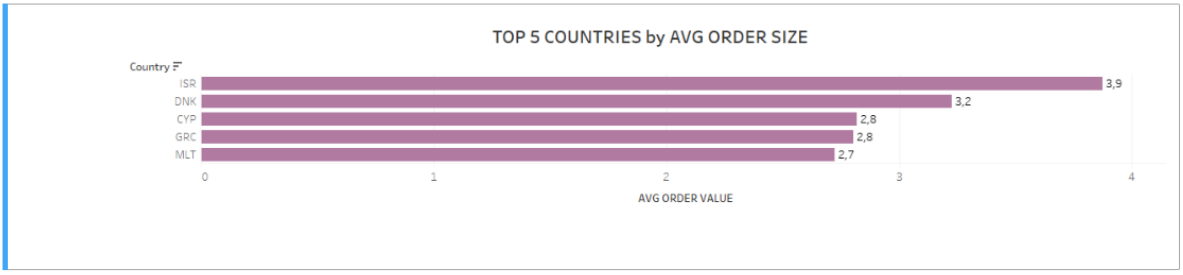
```sql
CREATE OR REPLACE VIEW PRODUCT_PROFIT_COUNTRY AS
SELECT PUI.PRODUCT_ID as PRODUCT_ID,
    PUI.VENUE_ID as VENUE_ID,
    P.COUNTRY AS COUNTRY,
     P.TIME_DELIVERED AS TIME_DELIVERED,
    PURCHASE_COUNT as PURCHASE_QUANTITY,
    PUI.PURCHASE_ID as PURCHASE_ID
    (COST_PER_UNIT_EUR*(1 + APPLICABLE_TAX_PERC )) as COST_PER_UNIT_EUR_VAT,
   (BASEPRICE)*(COST_PER_UNIT_EUR/COST_PER_UNIT) AS PRODUCT_PRICE_EUR
FROM PURCHASE_ITEM AS PUI
JOIN ITEM AS I
    ON PUI.PRODUCT_ID = I.PRODUCT_ID
JOIN PURCHASE AS P
    ON PUI.VENUE_ID = P.VENUE_ID
WHERE COST_PER_UNIT_EUR IS NOT NULL
AND PURCHASE_COUNT IS NOT NULL
AND BASEPRICE IS NOT NULL
GROUP BY COUNTRY

CREATE OR REPLACE VIEW TOTAL_PROFIT_EUR_COUNTRY AS
SELECT PURCHASE_ID,
        PRODUCT_ID,
        VENUE_ID,
        COUNTRY,
        TIME_DELIVERED,
        AVG(PRODUCT_PRICE_EUR) AS AVG_ORDER_VALUE
        AVG(PURCHASE_QUANTITY) AS AVG_PURCHASE_SIZE
        SUM(PRODUCT_PRICE_EUR)*SUM(PURCHASE_QUANTITY)/SUM(PURCHASE_QUANTITY) AS AOV
FROM PRODUCT_PROFIT_COUNTRY
GRoup by COUNTRY

--by avg order size
SELECT COUNTRY , AVG_PURCHASE_SIZE
FROM TOTAL_PROFIT_EUR_COUNTRY
GROUP BY COUNTRY
ORDER BY AVG_PURCHASE_SIZE desc Limit 5

--by avg order value
SELECT COUNTRY , AVG_ORDER_VALUE
FROM TOTAL_PROFIT_EUR_COUNTRY
GROUP BY COUNTRY
ORDER BY AVG_ORDER_VALUE desc Limit 5
```

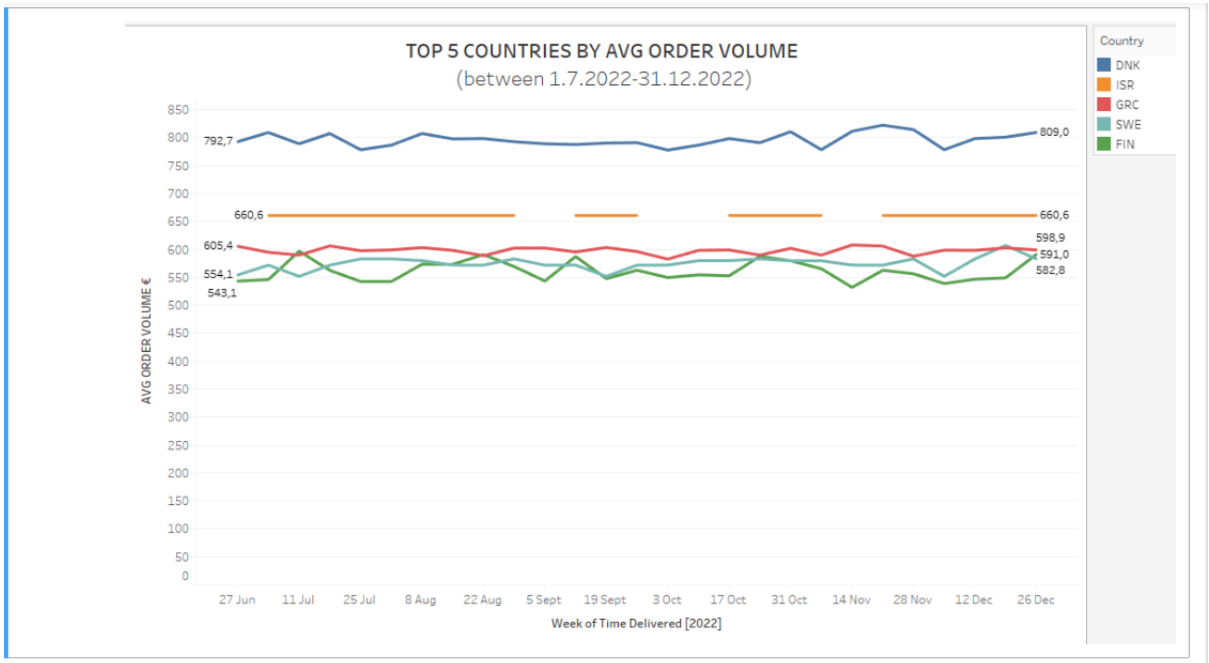
```sql
--by AOV
SELECT COUNTRY ,TIME_DELIVERED, AOV
FROM TOTAL_PROFIT_EUR_COUNTRY
GROUP BY COUNTRY
ORDER BY AOV desc Limit 5
```

VISUALIZATION









1. When creating the gross margin calculations, we need to calculate first the TOTAL REVENUE and TOTAL PROFIT. To get the revenue and profit, we use COST_PER_UNIT, COST_PER_UNIT_EUR, APPLICABLE_TAX, PURCHASE_COUNT and BASEPRICE values from our tables. We assume the costs and base price data are in local currencies or Euro. Applicable tax is in percentage and purchase count is integer. We assume these data are not null or !=0, otherwise they would mess up the calculations, therefore I excluded those from the calculations. According to the customer needs we can decide to modify the missing values (f.e. 0 or -999 for numeric and 'missing' for categorical data). In this case I decided to leave them as NULL.

2. There were missing values in most of the columns in our tables. I didn't insert rows where PRODUCT_ID OR PURCHASE_ID were missing. For the calculations I also excluded the NULL values of COSTS, BASEPRICES and TAX. I also encountered problems on my computer (not enough memory) because the files were too large. To be able to create visualization of the data with TABLEAU PUBLIC I used first 50 000 rows of each file. I know this is not the best solution, but because of the lack of time, this was the only way I could provide the visualizations and it gives as an idea about the TOP countries and venues.

3. To improve the solution, it would be useful to have names of the venues and description of the products, for better understanding of the purchases. From the VENUE ID and PRODUCT ID it is hard to recognize the top venues or the most sold products. TO solve the problem with the large data sets, we could split the data in to quartiles to smaller files or remove unnecessary columns.