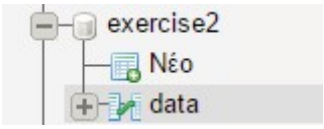
FINANCIAL INFORMATION SYSTEMS

Παναγιώτης Καψάλης DS3516005
pkapsalis@aueb.gr

Στα πλαίσια της δεύτερης εργασίας του μαθήματος κατέβασα από τον ιστότοπο τα δεδομένα, από αυτά κράτησα τις στήλες που αφορούσαν τον τύπο των μηνημάτων MsgType, το Outcome καθώς και την Ημερομηνία και ώρα που έγιναν τα αντίστοιχα FIX μηνύματα. Με την χρήση του κώδικα που υπάρχει στο αρχείο main.py έγινε η αποθήκευση των απαιτούμενων δεδομένων σε ένα csv αρχείο το οποίο στην συνέχεια θα φορτωθεί στην βάση δεδομένων. Για την δημιουργία της βάσης δεδομένων έγινε χρήση του WampServer.
Αρχικά πρέπει να δημιουργήσουμε μια βάση δεδομένων την οποία και ονομάζουμε exercise2, στην συνέχεια δημιουργούμε ένα πίνακα data με στήλες MsgType, MsgDate, Outcome, MsgTime, TradeId (id του trade που γινεται), TradeAmount (πλήθος μετοχών που αγοράστηκαν), TradePrice (το ποσό της συναλλαγής) για την αποθήκευση των στηλών του csv. Η δομή της database είναι η παρακάτω που βλεπουμε στην εικόνα:



Η δομή του πίνακα data στον οποίο θα καταχωρήσουμε τα records του csv είναι η παρακάτω:
Επιπλέον θέτουμε και την στήλη id η οποία έχει το χαρακτηριστικό autoincreament και σηματοδοτεί την ταυτότητα των records. Συνεπώς με την δημιουργία της βάσης δεδομένων και του πίνακα ολοκληρώσαμε τα πρώτα βήματα της εργασίας.

	#	Όνομα	Τύπος	Σύνθεση
☐	1	MsgType	char(5)	utf8_general_ci
☐	2	Outcome	text	latin1_swedish_ci
☐	3	MsgDate	text	latin1_swedish_ci
☐	4	MsgTime	text	latin1_swedish_ci
☐	5	id	int(11)	
☐	6	TradeId	text	latin1_swedish_ci
☐	7	TradeAmount	text	latin1_swedish_ci
☐	8	TradePrice	text	latin1_swedish_ci

Φόρτωση των δεδομένων στην βάση

Πρόκειται για περίπου 6 εκατομύρια entries οπότε ο πιο αποδοτικός τρόπος είναι μέσω SQL ο κώδικας που χρησιμοποιήθηκε είναι ο παρακάτω (αρχείο **loadCSV.sql**)

```
LOAD DATA INFILE 'C:/Users/User/Desktop/FIS/Exercise2/out2.csv' IGNORE
INTO TABLE data
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
```

Παρατίθεται screenshot της βάσης, πρόκειται για τις πρώτες 24 εγγραφές.

Εμφάνιση εγγραφών 0 - 24 (6411995 συνολικά, Το ερώτημα χρειάστηκε 0.0005 δευτερόλεπτα)

	MsgType	Outcome	MsgDate	MsgTime	TradeId	TradeAmount	TradePrice	id
Επεξεργασία Αντιγραφή Διαγραφή	"MsgT	"Outcome"	"MsgDate"	"MsgTime"	"TradeId"	"TradeAmount"	"TradePrice"	1
Επεξεργασία Αντιγραφή Διαγραφή	"5"	"1"	"2015-12-30"	"23:00:04.163"	"0"	"0"	"0"	2
Επεξεργασία Αντιγραφή Διαγραφή	"5"	"1"	"2015-12-30"	"23:00:03.510"	"0"	"0"	"0"	3
Επεξεργασία Αντιγραφή Διαγραφή	"5"	"1"	"2015-12-30"	"23:00:00.327"	"0"	"0"	"0"	4
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:59:52.777"	"0"	"0"	"0"	5
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:59:47.147"	"0"	"0"	"0"	6
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:59:42.620"	"0"	"0"	"0"	7
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:59:42.077"	"0"	"0"	"0"	8
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:59:41.343"	"0"	"0"	"0"	9
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:59:41.060"	"0"	"0"	"0"	10
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:59:22.670"	"0"	"0"	"0"	11
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:59:17.740"	"0"	"0"	"0"	12
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:59:12.623"	"0"	"0"	"0"	13
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:59:12.623"	"0"	"0"	"0"	14
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:59:11.343"	"0"	"0"	"0"	15
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:59:11.343"	"0"	"0"	"0"	16
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:58:52.530"	"0"	"0"	"0"	17
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:58:47.320"	"0"	"0"	"0"	18
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:58:42.623"	"0"	"0"	"0"	19
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:58:42.250"	"0"	"0"	"0"	20
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:58:41.343"	"0"	"0"	"0"	21
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:58:41.233"	"0"	"0"	"0"	22
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"0"	"2015-12-30"	"22:58:22.407"	"0"	"0"	"0"	23
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:58:17.913"	"0"	"0"	"0"	24
Επεξεργασία Αντιγραφή Διαγραφή	"0"	"1"	"2015-12-30"	"22:58:12.623"	"0"	"0"	"0"	25

Αφού γίνει η φόρτωση των δεδομένων θα συνεχίσουμε στην επεξεργασία των δεδομένων. Χρησιμοποιείται η SQL μέσω της γλώσσας προγραμματισμού Python έκδοση 3.5, για να επιτευχθεί αυτό εγκαθίσταται η βιβλιοθήκη cymysql, περισσότερες λεπτομέρειες υπάρχουν σε αυτόν εδώ τον σύνδεσμο για την συγκεκριμένη βιβλιοθήκη https://pypi.python.org/pypi/cymysql . Οπότε η σύνδεση με την βάση δεδομένων και της Python επιτυγχάνεται με τον παρακάτω κώδικα:

In [16]:
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import pandas as pd
import cymysql
import csv
import numpy as np
```

In [2]:
```
conn = cymysql.connect(host='127.0.0.1', user='root', passwd='', db='exercise2', charset='utf8')
cur = conn.cursor()
```

Η μέθοδος connect της cymysql λαμβάνει τις παραμέτρους της βάσης δεδομένων για να γίνει η

σύνδεση. Αφού επιτευχθεί η σύνδεση είμαστε έτοιμοι για την επεξεργασία των δεδομένων.

A) What is the total number of messages?

Γίνεται χρήση της συνάρτησης COUNT( ) της SQL για την καταμέτρηση όλων των entries της βάσης δεδομένων, ο παρακάτω κώδικας Python, χρησιμοποιεί τις συναρτήσης της cymysql δέχεται ως όρισμα τα SQL ερωτήματα και επιστρέφει τα αποτελέσματα από την βάση:

```
In [3]: ## 6.a What is the total number number of messages
        sql6a = """SELECT COUNT(id) FROM data"""
        cur.execute(sql6a)
        total_messages = cur.fetchone()[0]

In [50]: total_messages

Out[50]: 6411995
```

B) How many Heartbeat messages on average are transmitting per day?

Ουσιαστικά για την εύρεση των ζητούμενων, χρειάζεται να ομαδοποιήσουμε ανα μέρα τα μηνύματα με χαρακτηριστικό MsgType = 0, και για να βρούμε τον μέσο όρο να διαιρέσουμε με το πλήθος των groups. Αυτο επιτυγχάνεται με την χρήση του GROUP BY της SQL. Οπότε έχουμε:

```
In [4]: ###6.b How many heartbeat messages on average are transmitting per day
        sql6b = """SELECT MsgDate, COUNT(MsgType) FROM data WHERE MsgType LIKE '%0%' GROUP BY MsgDate"""
        cur.execute(sql6b)
        grouped_zeros = cur.fetchall()

In [10]: grouped_zeros
```

```
('"2014-01-06"', 8388),
('"2014-01-07"', 7346),
('"2014-01-08"', 6875),
('"2014-01-09"', 6868),
('"2014-01-10"', 7087),
('"2014-01-13"', 7099),
('"2014-01-14"', 7270),
('"2014-01-15"', 7283),
('"2014-01-16"', 7043),
('"2014-01-17"', 7606),
('"2014-01-20"', 7590),
('"2014-01-21"', 7748),
('"2014-01-22"', 7233),
('"2014-01-23"', 7871),
('"2014-01-24"', 7597),
('"2014-01-27"', 7174),
('"2014-01-28"', 7208),
('"2014-01-29"', 7141),
('"2014-01-30"', 7742),
```

```
In [11]: total_zero_messages = 0
         for item in grouped_zeros:
             total_zero_messages = total_zero_messages + item[1]
         print(total_zero_messages)

         4629270
```

Παράγουμε αρχικά τα groups, λίστα tuples και στην συνέχεια αρθροίζουμε το πλήθος των heartbeat messages. Το πλήθος όπως βλέπουμε παραπάνω είναι **4629270 messages.** Οπότε για να βρούμε το average διαιρούμε με το πλήθος των γκρουπ. 8417 είναι ο ζητούμενος average

```
In [17]: average_zero_messages = np.round(total_zero_messages/len(grouped_zeros))
         average_zero_messages

Out[17]: 8417.0
```

C) How many orders are in-coming per day?

```
In [52]: ##6.c How many new orders are incoming per days?
         sql6c = """SELECT MsgDate, Count(MsgType) FROM data WHERE Outcome LIKE '%0%' GROUP BY MsgDate"""
         cur.execute(sql6c)
         new_orders_per_day = cur.fetchall()
         new_orders_per_day

Out[52]: [('"2014-01-01"', 4192),
          ('"2014-01-02"', 4579),
          ('"2014-01-03"', 4926),
          ('"2014-01-06"', 4193),
          ('"2014-01-07"', 4812),
          ('"2014-01-08"', 4813),
          ('"2014-01-09"', 5772),
          ('"2014-01-10"', 5187),
          ('"2014-01-13"', 5224),
          ('"2014-01-14"', 5057),
          ('"2014-01-15"', 4832),
          ('"2014-01-16"', 4966),
          ('"2014-01-17"', 4991),
          ('"2014-01-20"', 4653),
          ('"2014-01-21"', 4738),
          ('"2014-01-22"', 5196),
          ('"2014-01-23"', 4286),
          ('"2014-01-24"', 4485),
          ('"2014-01-27"', 5499),
          ('"2014-01-28"', 4717)
```

Για τον υπολογισμό των incoming orders ανα μέρα γκρουπάρουμε τα μηνύματα τα οποία έχουν τις εξής τιμές στο πεδίο Outcome = 0. Πιο συγκεκριμένα ο κώδικας για τον υπολογισμό των groups είναι ο παρακάτω:

D) Which mothly period was the period with the most incoming orders and which with the less?

Εφόσον έχουμε τα records που αφορούν Incoming orders το μόνο που μένει είναι να τα επεξεργαστούμε για να πάρουμε το ζητούμενο αποτέλεσμα. Αρχικά από την παραπάνω λίστα των tuples παίρνουμε τις ημερομηνίες και οαμδοποιούμε τα μηνύματα ανάλογα με το έτος και το μήνα που πραγματοποιήθηκαν. Αφού γίνει αυτό βρίσκουμε ποια περίοδος (Έτος/Μήνας) είχε τα περισσότερα και λιγότερα incoming orders. Αρχικά μετατρέπουμε την ημερομηνία σε μορφή EEEE/MM και στην συνέχεια αποθηκεύουμε σε ένα dictionary τις ημερομηνίες ως κλειδιά, με σκοπό entries με ίδια κλειδία να αρθρόιζουν τις τιμές τους για τον υπολογισμό των incoming orders.

Στην συνέχεια με την χρήση της συνάρτησης min και max υπολογίζουμε και επιστρέφουμε το κλειδί, την χρονική περίοδο δηλαδή, και το πλήθος των Incoming orders. Στις παρακάτω εικόνες είναι τα αποσπάσματα κώδικα για την διαδικασία που περιγράφηκε.

Υπολογίστηκε ότι η χρονική περίοδος με τα περισσότερα incoming orders ήταν
2015-12 με 138951
και η χρονική περίοδος με τα λιγότερα incoming orders ήταν
2014-11 με 89270

In [53]:
```
#PYTHON-BASED APPROACH
##6.d Which mothly period was the period with the most incoming orders and which with the less
new_orders_per_day = [list(elem) for elem in new_orders_per_day]
for item in new_orders_per_day:
    item[0] = item[0].replace(' ', '')[:-4].upper()
new_orders_per_day
```

Out[53]:
```
[['"2014-01', 4192],
 ['"2014-01', 4579],
 ['"2014-01', 4926],
 ['"2014-01', 4193],
 ['"2014-01', 4812],
 ['"2014-01', 4813],
 ['"2014-01', 5772],
 ['"2014-01', 5187],
 ['"2014-01', 5224],
 ['"2014-01', 5057],
 ['"2014-01', 4832],
 ['"2014-01', 4966],
 ['"2014-01', 4991],
 ['"2014-01', 4653],
 ['"2014-01', 4738],
 ['"2014-01', 5196],
 ['"2014-01', 4286],
 ['"2014-01', 4485],
 ['"2014-01', 5499],
 ['"2014-01', 4717],
```

In [55]:
```
#PYTHON-BASED APPROACH
##6.d Which mothly period was the period with the most incoming orders and which with the less
new_orders = dict()
for item in new_orders_per_day:
    if item[0] not in new_orders.keys():
        new_orders[item[0]] = item[1]
    else:
        new_orders[item[0]] +=item[1]
new_orders
```

Out[55]:
```
{'"2014-01': 111602,
 '"2014-02': 94988,
 '"2014-03': 97862,
 '"2014-04': 111753,
 '"2014-05': 127131,
 '"2014-06': 125015,
 '"2014-07': 127350,
 '"2014-08': 107198,
 '"2014-09': 101805,
 '"2014-10': 110490,
 '"2014-11': 89270,
 '"2014-12': 107278,
 '"2015-01': 109887,
 '"2015-02': 99830,
 '"2015-03': 102643,
 '"2015-04': 105555,
 '"2015-05': 101409,
 '"2015-06': 102582,
 '"2015-07': 115050,
 '"2015-08': 133826,
 '"2015-09': 138241,
 '"2015-10': 138355,
 '"2015-11': 130861,
 '"2015-12': 138951}
```

In [56]:
```
maximum_incoming_orders = max(new_orders, key=new_orders.get)  # Just use 'min' instead of 'ma
```

```
In [56]: maximum_incoming_orders = max(new_orders, key=new_orders.get)  # Just use 'min' instead of 'max' for minimum.
         print(maximum_incoming_orders, new_orders[maximum_incoming_orders])

         "2015-12 138951
```

```
In [57]: minimum_incoming_orders = min(new_orders, key=new_orders.get)  # Just use 'min' instead of 'max' for minimum.
         print(minimum_incoming_orders, new_orders[minimum_incoming_orders])

         "2014-11 89270
```

E) What is the average number of trades that are traded every day?

Για να βρούμε το ζητούμενο γκρουπάρουμε ανα μέρα τα μηνύματα που έχουν χαρακτηριστικό MsgType = 8 και για τον υπολογισμό του μέσου όρου διαιρούμε με το πλήθος των groups.

```
In [44]: ##6.e What is the average number of trades, MsgType=8, that are traded every day
         sqle = """SELECT MsgDate, Count(MsgType) FROM data WHERE MsgType LIKE '%8%' GROUP BY MsgDate"""
         cur.execute(sqle)
         trades = cur.fetchall()
         trades
```

```
Out[44]: [('"2014-01-02"', 2213),
          ('"2014-01-03"', 2829),
          ('"2014-01-06"', 1),
          ('"2014-01-07"', 4499),
          ('"2014-01-08"', 5024),
          ('"2014-01-09"', 6698),
          ('"2014-01-10"', 4192),
          ('"2014-01-13"', 5263),
          ('"2014-01-14"', 5111),
          ('"2014-01-15"', 5361),
          ('"2014-01-16"', 4938),
          ('"2014-01-17"', 2901),
          ('"2014-01-20"', 2382),
          ('"2014-01-21"', 2291),
          ('"2014-01-22"', 4312),
          ('"2014-01-23"', 2119),
          ('"2014-01-24"', 3293),
          ('"2014-01-27"', 5404),
          ('"2014-01-28"', 4590),
```

Αφού έχει γίνει ο υπολογισμός των γκρουπ, απομένει η άρθροιση κατά γκρούπ και η διαίρεση του αποτελέσματος αυτού με το πλήθος των γκρουπ.

```
In [46]: total_trade_messages = 0
         for item in trades:
             total_trade_messages = total_trade_messages + item[1]
         print(total_trade_messages)

         1389206
```

```
In [49]: average_trades = np.round(total_trade_messages/len(trades))
         average_trades

Out[49]: 2829.0
```

Οπότε ο μέσος αριθμός trade μηνυμάτων είναι 2829.

F)
Για το συγκεκριμένο ερώτημα γκρουπάρουμε κατά MsgDate και TradeId, όπου το MsgType = 8, στο επόμενο βήμα εφαρμόζω καρτεσιανό γινόμενο του υπολογισθέν διανύσματος, με τον αρχικό πίνακα data. Από το αποτέλεσμα αυτού πάλι θα γκρουπάρουμε ανά μέρα και θα κάνουμε select το γινόμενο του (TradeAmount*TradePrice). Οπότε έχουμε ως εξής:

Αρχικά στην SQL κονσόλα του WampServer υπολογίζουμε τον temp πίνακα t1 ο οποίος υπολογίζει την μέγιστη ώρα ανά μέρα και επιστρέφει το TradeId, TradeAmount, TradePrice. Ο SQL κώδικας που χρησιμοποιήθηκε είναι ο παρακάτω για τον υπολογισμό του πίνακα t1:

```
SELECT data.MSgTime, data.MsgDate, data.TradeId, data.TradeAmount, data.TradePrice FROM
(SELECT max(MsgTime) as max_ts,MsgDate, TradeId FROM data WHERE MsgType LIKE
'%8%' GROUP BY MsgDate, TradeId) as T
INNER JOIN data
ON T.max_ts = data.MsgTime and T.MsgDate = data.MsgDate and T.TradeId = data.TradeId
```

Στην συνέχεια αφού υπολογίσουμε τον πίνακα t1 θα πάμε σε αυτόν και θα υπολογίσουμε το άρθροισμα του γινομένου (TradeAmount*TradePrice) και θα το γκρουπάρουμε ανά μήνα και έτος

Σημείωση: Έχω διαμορφώσει το παραπάνω sql ερώτημα, αλλά έτρεχε για παρα πολύ ώρα

Οδηγίες

1) Εκτέλεση του αρχείου main.py για την δημιουργία του αρχείου csv.
2) Στο αρχείο loadCSV.sql υπάρχει ο κώδικας σε SQL, που χρησιμοποιήθηκε για το γέμισμα του πίνακα data της βάσης δεδομένων που δημιούργησα.
3) Εκτέλεση του αρχείου code.py για την εκτέλεση του κώδικα που απαντά στα ερωτήματα που ζητήθηκαν και τεκμηριώθηκαν στην αναφορά.
4) Το αρχείο 6.f περιέχει το κώδικα sql για το ερώτημα 6.f