|  |
| --- |
| ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ |
| ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ |



ΠΡΟΤΖΕΚΤ ΣΤΑ ΠΛΑΙΣΙΑ ΤΟΥ ΕΡΓΑΣΤΗΡΙΟΥ ΤΟΥ ΜΑΘΗΜΑΤΟΣ

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

…

|  |  |  |
| --- | --- | --- |
| Όνομα | ΣΤΑΜΑΤΗΣ | ΧΡΙΣΤΙΝΑ-ΕΛΕΑΝΝΑ |
| Επώνυμο | ΚΑΖΑΣ | ΣΑΜΑΡΑ |
| Έτος | 3ο | 3ο |
| ΑΜ | 1084636 | 1084622 |

Ημερομηνία Παράδοσης:

Περιεχόμενα

[Κεφάλαιο 1: Δημιουργία Βάσης Δεδομένων 4](#_Toc126619019)

[1.1 Σχεσιακό Διάγραμμα 4](#_Toc126619020)

[1.2 Επεξήγηση 4](#_Toc126619021)

[1.3 Κώδικας sql για Create – Insert 5](#_Toc126619022)

[1.4 Πολλαπλές εισαγωγές κρατήσεων προσφορών 22](#_Toc126619023)

[Κεφάλαιο 2: Stored Procedures 23](#_Toc126619024)

[2.1 new\_driver() 23](#_Toc126619025)

[2.2 branch\_trip\_details() 25](#_Toc126619026)

[2.3 delete\_worker() 26](#_Toc126619027)

[3.4 28](#_Toc126619028)

[Κεφάλαιο 3: Triggers 29](#_Toc126619029)

[3.1 log updates 29](#_Toc126619030)

[3.2 update\_trip() 29](#_Toc126619031)

[3.3 changing\_salary() 31](#_Toc126619032)

[Κεφάλαιο 4: GUI 32](#_Toc126619033)

[4.1 Κώδικας Java για Insert 32](#_Toc126619034)

[4.2 Κώδικας Java για ζητούμενο εισαγωγής νέου υπεύθυνου πληροφορικής με ίσα δικαιώματα 33](#_Toc126619035)

[4.3 Κώδικας Java για ζητούμενο σχετικά με πληροφορίες υποκαταστήματος, διευθυντή υποκαταστήματος, σύνολο κρατήσεων και σύνολο εσόδων 34](#_Toc126619036)

[4.4 Κώδικας Java για ζητούμενο σχετικά ονοματεπώνυμα υπαλλήλων, και πληροφορίες μισθών 34](#_Toc126619037)

[4.5 Σύνδεση 36](#_Toc126619038)

[4.6 Εισαγωγή Εγγραφής 37](#_Toc126619039)

[4.7 Δικαιώματα Νέου IT Υπαλλήλου 41](#_Toc126619040)

[4.8 Ενημέρωση Εγγραφής 43](#_Toc126619041)

[4.9 Διαγραφή Εγγραφής 47](#_Toc126619042)

[4.10 Ζητούμενο για προβολή λεπτομερειών ταξιδιού 51](#_Toc126619043)

[4.11 Ζητούμενο για αναζήτηση κρατήσεων 53](#_Toc126619044)

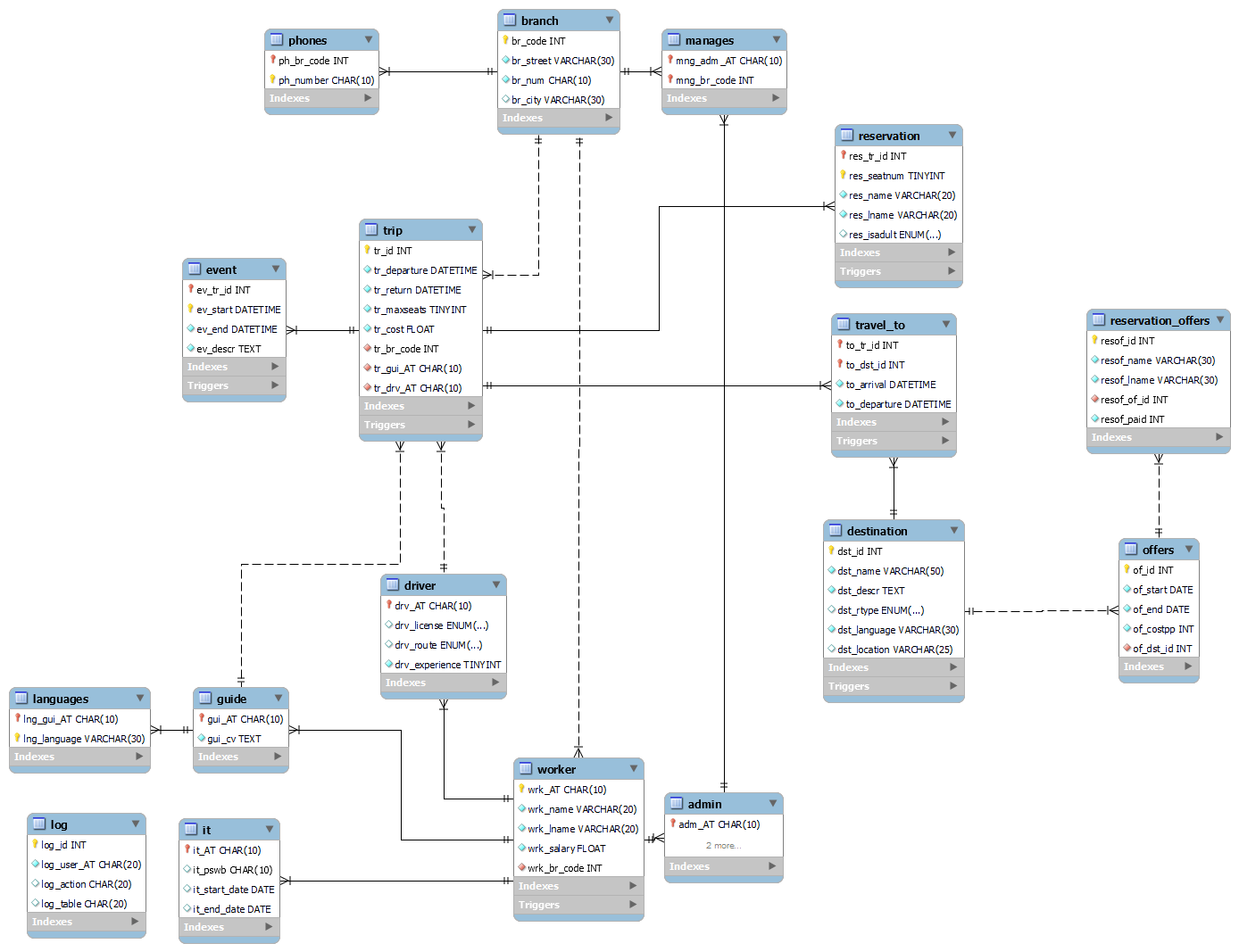
[4.12 Ζητούμενο για ένδειξη συνόλου κρατήσεων και εσόδων 55](#_Toc126619045)

[4.13 Ζητούμενο για ένδειξη πληροφοριών σχετικά με το υποκατάστημα και τους μισθούς 56](#_Toc126619046)

[4.17 EXTRA ζητούμενα 58](#_Toc126619047)

# Κεφάλαιο 1: Δημιουργία Βάσης Δεδομένων

## 1.1 Σχεσιακό Διάγραμμα



## 1.2 Επεξήγηση

Παραπάνω βλέπουμε το σχεσιακό διάγραμμα της βάσης δεδομένων μας (travel\_agency). Η βάση αποτελείται από 17 πίνακες που προσομοιάζουν την λειτουργία ενός γραφείου ταξιδιών.

Πιο συγκεκριμένα, ο πίνακας **branch** αναπαριστά τα υποκαταστήματα του γραφείου, τα τηλέφωνα των οποίων βρίσκονται στον πίνακα **phones**. Έπειτα έχουμε τον πίνακα **worker**, στον οποίο είναι αποθηκευμένοι όλοι οι εργαζόμενοι όλων των υποκαταστημάτων. Κάθε εγγραφή ενός worker περιέχει μια αναφορά του υποκαταστήματος στο οποίο εργάζεται. Οι workers χωρίζονται σε διοικητικούς (**admin**), οδηγούς (**driver**), ξεναγούς (**guide**) και υπεύθυνους πληροφορικής (**it**). Ο πίνακας guide συσχετίζεται με τον πίνακα **languages**, ο οποίος δείχνει ποιες γλώσσες μιλάει κάθε ξεναγός. Επίσης οι διοικητικοί συσχετίζονται με τα υποκαταστήματα μέσω του πίνακα **manages**, ο οποίος παρουσιάζει τους διευθυντές των υποκαταστημάτων. Ο πίνακας **trip** αναπαριστά τα ταξίδια που έχουν διοργανωθεί, καθώς και όλες τις πληροφορίες τους, όπως το υποκατάστημα που τις διοργανώνει, τον οδηγό και τον ξεναγό τους. Τα ταξίδια συσχετίζονται και με τον πίνακα **events**, στον οποίο αναπαρίστανται οι σημαντικότερες εκδηλώσεις κάθε trip. Ακόμη, υπάρχει ο πίνακας **destination**, ο οποίος περιλαμβάνει όλους τους δυνατούς προορισμούς των ταξιδιών. Οι προορισμοί συσχετίζονται με τα ταξίδια με τον πίνακα **travel\_to**, ο οποίος δείχνει τον προορισμό κάθε ταξιδιού. Ο πίνακας **reservation** περιλαμβάνει τις κρατήσεις που έχουν γίνει για κάθε ταξίδι. Επιπλέον, έχουμε τους πίνακες **offers** και **reservation\_offers**, οι οποίοι περιγράφουν τα μελλοντικά ταξίδια που θα πραγματοποιηθούν και τις κρατήσεις σε αυτά αντίστοιχα. Τέλος, έχουμε τον πίνακα **log**, ο οποίος αποτελεί πίνακα καταγραφής ενεργειών, ο οποίος ενημερώνεται σε κάθε ενέργεια (insert, update, delete) που εκτελείται σε ορισμένους πίνακες. Ο πίνακας log δεν συσχετίζεται με κανέναν άλλον πίνακα της βάσης. Η ενημέρωσή του γίνεται με triggers, τα οποία θα περιγράφουν την συνέχεια.

## 1.3 Κώδικας sql για Create – Insert

DROP DATABASE IF EXISTS travel\_agency;

CREATE DATABASE IF NOT EXISTS travel\_agency;

USE travel\_agency;

CREATE TABLE branch(

br\_code INT NOT NULL AUTO\_INCREMENT,

br\_street VARCHAR(30) NOT NULL,

br\_num CHAR(10) NOT NULL,

br\_city VARCHAR(30),

PRIMARY KEY (br\_code)

);

CREATE TABLE worker(

wrk\_AT CHAR(10) UNIQUE NOT NULL,

wrk\_name VARCHAR(20) NOT NULL,

wrk\_lname VARCHAR(20) NOT NULL,

wrk\_salary FLOAT NOT NULL,

wrk\_br\_code INT NOT NULL,

PRIMARY KEY (wrk\_AT),

CONSTRAINT WRKBR FOREIGN KEY (wrk\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE it(

it\_AT CHAR(10) UNIQUE NOT NULL,

it\_pswb CHAR(10) DEFAULT "password",

it\_start\_date DATE,

it\_end\_date DATE,

primary key(it\_AT),

CONSTRAINT ITWRK FOREIGN KEY (it\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE driver(

drv\_AT CHAR(10) UNIQUE NOT NULL,

drv\_license ENUM("A", "B", "C", "D"),

drv\_route ENUM("LOCAL", "ABROAD"),

drv\_experience TINYINT NOT NULL,

PRIMARY KEY (drv\_AT),

CONSTRAINT DRWRK FOREIGN KEY (drv\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE guide(

gui\_AT CHAR(10) UNIQUE NOT NULL,

gui\_cv TEXT NOT NULL,

PRIMARY KEY (gui\_AT),

CONSTRAINT GUWRK FOREIGN KEY (gui\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE admin(

adm\_AT CHAR(10) UNIQUE NOT NULL,

adm\_type ENUM("LOGISTICS", "ADMINISTRATIVE", "ACCOUNTING"),

adm\_diploma VARCHAR(200) NOT NULL,

PRIMARY KEY (adm\_AT),

CONSTRAINT ADMWRK FOREIGN KEY (adm\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE manages(

mng\_adm\_AT CHAR(10) UNIQUE NOT NULL,

mng\_br\_code INT NOT NULL,

PRIMARY KEY (mng\_adm\_AT, mng\_br\_code),

CONSTRAINT MNGWRK FOREIGN KEY (mng\_adm\_AT) REFERENCES admin(adm\_AT)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT MNGBR FOREIGN KEY (mng\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE trip (

tr\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,

tr\_departure DATETIME NOT NULL,

tr\_return DATETIME NOT NULL,

tr\_maxseats TINYINT NOT NULL,

tr\_cost FLOAT NOT NULL,

tr\_br\_code INT NOT NULL,

tr\_gui\_AT CHAR(10) NOT NULL,

tr\_drv\_AT CHAR(10) NOT NULL,

PRIMARY KEY (tr\_id),

CONSTRAINT TRBR FOREIGN KEY (tr\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT TRDR FOREIGN KEY (tr\_drv\_AT) REFERENCES driver(drv\_AT)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT TRGU FOREIGN KEY (tr\_gui\_AT) REFERENCES guide(gui\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE event (

ev\_tr\_id INT NOT NULL AUTO\_INCREMENT,

ev\_start DATETIME NOT NULL,

ev\_end DATETIME NOT NULL,

ev\_descr TEXT NOT NULL,

PRIMARY KEY (ev\_tr\_id , ev\_start),

CONSTRAINT EVTR FOREIGN KEY (ev\_tr\_id) REFERENCES trip (tr\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE phones(

ph\_br\_code INT NOT NULL,

ph\_number CHAR(10) NOT NULL,

PRIMARY KEY (ph\_br\_code, ph\_number),

CONSTRAINT PHBR FOREIGN KEY (ph\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE languages (

lng\_gui\_AT CHAR(10) NOT NULL,

lng\_language VARCHAR(30) NOT NULL,

PRIMARY KEY (lng\_gui\_AT, lng\_language),

CONSTRAINT LNGUI FOREIGN KEY (lng\_gui\_AT) REFERENCES guide(gui\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE destination(

dst\_id INT UNIQUE NOT NULL AUTO\_INCREMENT,

dst\_name VARCHAR(50) NOT NULL,

dst\_descr TEXT NOT NULL,

dst\_rtype ENUM("LOCAL", "ABROAD"),

dst\_language VARCHAR(30) NOT NULL,

dst\_location VARCHAR(30),

PRIMARY KEY (dst\_id)

);

CREATE TABLE travel\_to(

to\_tr\_id INT UNIQUE NOT NULL,

to\_dst\_id INT NOT NULL,

to\_arrival DATETIME NOT NULL,

to\_departure DATETIME NOT NULL,

PRIMARY KEY (to\_tr\_id, to\_dst\_id),

CONSTRAINT TOTR FOREIGN KEY (to\_tr\_id) REFERENCES trip(tr\_id)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT TODST FOREIGN KEY (to\_dst\_id) REFERENCES destination(dst\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE reservation(

res\_tr\_id INT NOT NULL,

res\_seatnum TINYINT NOT NULL,

res\_name VARCHAR(20) NOT NULL,

res\_lname VARCHAR(20) NOT NULL,

res\_isadult ENUM("ADULT", "MINOR"),

PRIMARY KEY (res\_tr\_id, res\_seatnum),

CONSTRAINT RESTR FOREIGN KEY (res\_tr\_id) REFERENCES trip(tr\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE offers(

of\_id INT UNIQUE NOT NULL,

of\_start DATE NOT NULL,

of\_end DATE NOT NULL,

of\_costpp INT NOT NULL,

of\_dst\_id INT NOT NULL,

primary key(of\_id),

CONSTRAINT OFDST FOREIGN KEY (of\_dst\_id) REFERENCES destination(dst\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE reservation\_offers(

resof\_id INT UNIQUE NOT NULL AUTO\_INCREMENT,

resof\_name VARCHAR(30) NOT NULL DEFAULT "unknown",

resof\_lname VARCHAR(30) NOT NULL DEFAULT "unknown",

resof\_of\_id INT NOT NULL,

resof\_paid INT NOT NULL,

primary key(resof\_id),

CONSTRAINT RESOFOF FOREIGN KEY (resof\_of\_id) REFERENCES offers(of\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE log(

log\_id INT UNIQUE NOT NULL AUTO\_INCREMENT,

log\_user\_AT CHAR(20) NOT NULL,

log\_action CHAR(20),

log\_table CHAR(20),

primary key(log\_id)

);

CREATE TABLE temp\_table(

resof\_name VARCHAR(30) NOT NULL DEFAULT "unknown",

resof\_lname VARCHAR(30) NOT NULL DEFAULT "unknown");

INSERT INTO branch VALUES

(1, 'Themistokleoys', 56, 'Thessaloniki'),

(NULL, 'Kyklopon', 32, 'Athina'),

(NULL, 'Timonos', 164, 'Athina'),

(NULL, 'Pallinaion', 5, 'Sparti'),

(NULL, 'Afidnaion', 66, 'Volos'),

(NULL, 'Rodopis', 71, 'Thessaloniki'),

(NULL, 'Grevenon', 24, 'Athina'),

(NULL, 'Fokou', 80, 'Volos'),

(NULL, 'Persefonis', 100, 'Thessaloniki'),

(NULL, 'Agisilaou', 2, 'Athina'),

(NULL, "Rizospastwn", 35, "Argostoli"),

(NULL, "Bergwth", 43, "Athens"),

(NULL, "Momferatou", 8, "Thessaloniki"),

(NULL, "Aetoraxis", 77, "Larisa"),

(NULL, "Byronws", 53, "Arta"),

(NULL, "Skoufa", 43, "Kerkyra"),

(NULL, "Pyrrou", 17, "Hrakleio"),

(NULL, "Antwnh Tritsh", 82, "Tripoli"),

(NULL, "Sitemporwn", 24, "Florina"),

(NULL, "Devosetou", 47, "Skiathos");

INSERT INTO worker VALUES

('ATK1315846', 'Petros', 'Kazatzis', 1000.00, 1),

('ATK4836520', 'Giorgos', 'Kalamis', 1500.50, 2),

('AMN0046398', 'Katerina', 'Vasilioy', 2000.00, 3),

('ATK8477656', 'Kostas', 'Papoytsis', 1300.00, 4),

('AMN3526783', 'Maria', 'Peponi', 1200.30, 5),

('BGK8462000', 'Spyros', 'Karboyniaris', 1700.00, 6),

('AMN7429291', 'Sokratis', 'Voyrdas', 2000.50, 7),

('BGK8881231', 'Stelios', 'Papapetroy', 1000.00, 8),

('ATK7776639', 'Konstantina', 'Karpatsioy', 1200.00, 9),

('BGK3727188', 'Perseas', 'Kallikraths', 2200.00, 10),

('ATK4638999', 'Patroklos', 'Kostopoylos', 900.00, 1),

('AMN1119994', 'Leonidas', 'Nikolaidis', 1100.00, 2),

('AMN3647589', 'Persofoni', 'Antipa', 1350.40, 3),

('BGK0001921', 'Pelopidas', 'Ologios', 1550.00, 4),

('BGK4322253', 'Ermionh', 'Feroypi', 1600.00, 5),

('AMN3909461', 'Periklis', 'Tertaros', 1300.00, 6),

('BGK3343351', 'Georgia', 'Xenioy', 1380.00, 7),

('ATK9354465', 'Sotiria', 'Deligioy', 1100.00, 8),

('ATK5243784', 'Mixalis', 'Alvakis', 990.00, 9),

('BGK6458247', 'Elpida', 'Ferosoy', 1050.00, 10),

('AMN5182954', 'Giorgos', 'Violis', 1230.00, 1),

('BGK2743822', 'Maria', 'Troykoy', 980.00, 2),

('ATK7842587', 'Giota', 'Sparoy', 1000.00, 3),

('BGK1313478', 'Petros', 'Mixoy', 890.00, 4),

('BGK4782387', 'Kostas', 'Kostidis', 980.00, 5),

('ATK4388828', 'Achilleas', 'Terteris', 980.00, 6),

('BGK7473628', 'Ektoras', 'Lastos', 1240.50, 7),

('AMN9090999', 'Dimitra', 'Zenisi', 1000.00, 8),

('BGK7483990', 'Athina', 'Eromaxi', 960.00, 9),

('ATK7788877', 'Aris', 'Spathostratos', 1450.00, 10),

("AT435342", "Michael", "Scott", 3500, 12),

("AT647865", "Jim", "Halpert", 2100, 13),

("AT432356", "Dwight", "Schrute", 2100, 15),

("AT546785", "Pam", "Beesly", 1500, 13),

("AT647392", "Andy", "Bernard", 2200, 14),

("AT930291", "Stanley", "Hudson", 1900, 17),

("AT849382", "Phyllis", "Vance", 2050, 18),

("AT253647", "Angela", "Martin", 2200, 19),

("AT125321", "Oscar", "Martinez", 2200, 16),

("AT986695", "Kevin", "Malone", 1700, 11),

("AT541121", "David", "Wallas", 4000, 11),

("AT994032", "Meredith", "Palmer", 1900, 12),

("AT554332", "Creed", "Bratton", 1650, 14),

("AT885736", "Ryan", "Howard", 1500, 15),

("AT657483", "Robert", "California", 2600, 11),

("AT439233", "Kelly", "Kapoor", 2000, 12),

("AT453890", "Toby", "Flenderson", 2300, 13),

("AT899878", "Darryl", "Phibbin", 1800, 14),

("AT675038", "Roy", "Anderson", 1800, 15),

("AT533234", "Erin", "Hannon", 1750, 16),

("AT903827", "Gabe", "Lewis", 2300, 17),

("AT758483", "Clark", "Green", 2100, 18),

("AT234955", "Pete", "Miller", 2250, 19),

("AT857362", "Jo", "Bennett", 4000, 20),

("AT749920", "Rachel", "Green", 2100, 20),

("AT434253", "Monica", "Geller", 2200, 20),

("AT909583", "Phoebe", "Buffay", 1900, 16),

("AT640901", "Joey", "Tribbiani", 2000, 17),

("AT125432", "Chandler", "Bing", 2400, 14),

("AT789504", "Ross", "Geller", 1750, 12),

("ATK94738", "Vasiliki", "Giotoy", 1500, 1);

INSERT INTO driver VALUES

('ATK4388828', 'A', 'ABROAD', 5),

('BGK7473628', 'C', 'LOCAL', 10),

('AMN9090999', 'A', 'LOCAL', 3),

('BGK7483990', 'B', 'LOCAL', 9),

('ATK7788877', 'D', 'ABROAD', 15),

('AMN5182954', 'B', 'LOCAL', 5),

('BGK2743822', 'D', 'ABROAD', 13),

('ATK7842587', 'A', 'ABROAD', 20),

('BGK1313478', 'B', 'ABROAD', 17),

('BGK4782387', 'C', 'LOCAL', 1),

("AT435342", "A", "LOCAL", 5),

("AT647865", "B", "ABROAD", 8),

("AT432356", "C", "LOCAL", 2),

("AT546785", "D", "ABROAD", 6),

("AT647392", "A", "ABROAD", 7),

("AT930291", "B", "LOCAL", 3),

("AT849382", "C", "ABROAD", 1),

("AT253647", "D", "LOCAL", 4),

("AT125321", "A", "LOCAL", 9),

("AT986695", "B", "ABROAD", 5);

INSERT INTO admin VALUES

('ATK1315846', 'LOGISTICS', 'logistics'),

('ATK4836520', 'ADMINISTRATIVE', 'management'),

('AMN0046398', 'ACCOUNTING', 'economic'),

('ATK8477656', 'ACCOUNTING', 'economic'),

('AMN3526783', 'LOGISTICS', 'logistics'),

('BGK8462000', 'LOGISTICS', 'logistics'),

('AMN7429291', 'ADMINISTRATIVE', 'management'),

('BGK8881231', 'LOGISTICS', 'logistics'),

('ATK7776639', 'ACCOUNTING', 'economic'),

('BGK3727188', 'LOGISTICS', 'logistics'),

("AT903827", "ADMINISTRATIVE", "Master in International Trade Relations"),

("AT758483", "ADMINISTRATIVE", "Master in Climate Change, Development and POlicy"),

("AT234955", "ADMINISTRATIVE", "Master in Diplomacy and International Organizations"),

("AT857362", "LOGISTICS", "Master in International Supply Chain Management and Logistics"),

("AT749920", "LOGISTICS", "PhD in Bussiness Economics"),

("AT434253", "LOGISTICS", "PhD in Economics"),

("AT909583", "LOGISTICS", "Master in Applied Mathematics"),

("AT640901", "ACCOUNTING", "Master in Finance, Accounting and Taxation"),

("AT125432", "ACCOUNTING", "Master in Accounting and Bussiness Information Technology"),

("AT789504", "ACCOUNTING", "Master in Forensic Accounting");

INSERT INTO manages VALUES

('ATK1315846', 1),

('ATK4836520', 2),

('AMN0046398', 3),

('ATK8477656', 4),

('AMN3526783', 5),

('BGK8462000', 6),

('AMN7429291', 7),

('BGK8881231', 8),

('ATK7776639', 9),

('BGK3727188', 10),

("AT903827", 11),

("AT758483", 12),

("AT234955", 13),

("AT857362", 14),

("AT749920", 15),

("AT434253", 16),

("AT909583", 17),

("AT640901", 18),

("AT125432", 19),

("AT789504", 20);

INSERT INTO phones VALUES

(1, '6984637882'),

(2, '6983662711'),

(3, '8974466020'),

(4, '6980023131'),

(5, '6973232000'),

(6, '6981212343'),

(7, '6975489377'),

(8, '6937584990'),

(9, '6973918273'),

(10, '6938847726'),

(11, "2671025225"),

(12, "2071093847"),

(13, "3471098732"),

(14, "5471083982"),

(15, "3271083943"),

(16, "9847382746"),

(17, "8738293884"),

(18, "7637266372"),

(19, "4885935704"),

(20, "7839283746");

INSERT INTO guide VALUES

('ATK4638999', 'Istorikos'),

('AMN1119994', 'Archaiologos'),

('AMN3647589', 'Istorikos'),

('BGK0001921', 'Istorikos'),

('BGK4322253', 'Archaiologos'),

('AMN3909461', 'Theologos'),

('BGK3343351', 'Moysikos'),

('ATK9354465', 'Theologos'),

('ATK5243784', 'Archaiologos'),

('BGK6458247', 'Istorikos'),

("AT541121", "Master in History of Ancient Civilations"),

("AT994032", "Master in Archaeology"),

("AT554332", "Phd in Paleontology"),

("AT885736", "Master in Mediterranean Archaeology"),

("AT657483", "Philosopher"),

("AT439233", "Diploma in Human Osteoarchaeology"),

("AT453890", "Diploma in European Archaeology"),

("AT899878", "Phd in Historical Culture and Medieval Worlds"),

("AT675038", "Diploma in Advanced Paleontology"),

("AT533234", "Master of Arts in Humanities");

INSERT INTO languages VALUES

('ATK4638999', 'Agglika'),

('AMN1119994', 'Ispanika'),

('AMN3647589', 'Mandarinika'),

('BGK0001921', 'Germanika'),

('BGK4322253', 'Italika'),

("AT541121", "English, Greek, Spanish"),

("AT994032", "Greek, Arabic, Italian, German"),

("AT554332", "Danish, English, French"),

("AT885736", "French, Polish, Norwegian"),

("AT657483", "English, Chinese, Hindi"),

("AT439233", "French, Portuguese, Spanish"),

("AT453890", "German, Portuguese, Greek"),

("AT899878", "Japanese, Dutch, Czech"),

("AT675038", "Greek, Italian, French"),

("AT533234", "German, English, Greek");

INSERT INTO trip VALUES

(1, "2022-09-03 17:30:00", "2022-09-09 18:00:00", 100, 750.00, 1, 'ATK4638999', 'AMN5182954'),

(2, "2022-07-10 08:00:00", "2022-07-15 08:00:00", 120, 1000.00, 2, 'AMN1119994', 'BGK2743822'),

(3, "2022-11-15 09:00:00", "2022-11-17 15:00:00", 75, 360.00, 3, 'AMN3647589', 'ATK7842587'),

(4, "2022-09-01 06:00:00", "2022-09-08 18:00:00", 100, 1200.00, 4, 'BGK0001921', 'BGK1313478'),

(5, "2022-02-02 11:00:00", "2022-02-05 05:00:00", 50, 950.00, 5, 'BGK4322253', 'BGK4782387'),

(6, "2022-01-13 13:00:00", "2022-01-19 13:00:00", 100, 880.00, 6, 'AMN3909461', 'ATK4388828'),

(7, "2023-05-22 04:00:00", "2023-05-27 04:00:00", 60, 1350.00, 7, 'BGK3343351', 'BGK7473628'),

(8, "2023-02-10 14:00:00", "2023-03-01 17:00:00", 90, 2200.00, 8, 'ATK9354465', 'AMN9090999'),

(9, "2023-08-11 19:00:00", "2023-08-13 09:00:00", 35, 1100.00, 9, 'ATK5243784', 'BGK7483990'),

(10, "2023-11-11 07:00:00", "2023-11-18 10:00:00", 90, 890.00, 10, 'BGK6458247', 'ATK7788877'),

(11, "2023-02-03 9:00:00", "2023-02-09 19:00:00", 1, 750.00, 11, "AT541121", "AT435342"),

(12, "2023-03-07 9:00:00", "2023-03-11 20:00:00", 120, 500.00, 12, "AT994032", "AT647865"),

(13, "2023-04-02 9:00:00", "2023-04-08 11:00:00", 75, 600.00, 13, "AT554332", "AT432356"),

(14, "2023-05-01 9:00:00", "2023-05-06 16:00:00", 90, 550.00, 14, "AT885736", "AT546785"),

(15, "2023-06-08 9:00:00", "2023-06-13 13:00:00", 120, 650.00, 15, "AT657483", "AT647392"),

(16, "2023-07-09 9:00:00", "2023-07-16 05:00:00", 90, 820.00, 16, "AT439233", "AT930291"),

(17, "2023-08-10 9:00:00", "2023-08-19 15:00:00", 110, 1000.00, 17, "AT453890", "AT849382"),

(18, "2023-09-12 9:00:00", "2023-09-20 08:00:00", 80, 950.00, 18, "AT899878", "AT253647"),

(19, "2023-11-17 9:00:00", "2023-11-23 20:00:00", 100, 470.00, 19, "AT675038", "AT125321"),

(20, "2023-12-20 9:00:00", "2023-12-30 03:00:00", 120, 1500.00, 20, "AT533234", "AT986695");

INSERT INTO event VALUES

(1, "2022-09-04 12:00:00", "2022-09-04 14:00:00", "Pop Party"),

(2, "2022-07-12 17:00:00", "2022-07-12 23:00:00", "Rave Party"),

(3, "2022-11-15 19:00:00", "2022-11-15 22:00:00", "Concert Adele"),

(4, "2022-09-02 10:00:00", "2022-09-02 12:00:00", "Formula"),

(5, "2022-02-02 15:00:00", "2022-02-02 19:00:00", "Chess Tournament"),

(6, "2022-01-14 09:00:00", "2022-01-14 13:00:00", "Bird Watching"),

(7, "2023-05-22 19:00:00", "2023-05-22 23:00:00", "TommorowLand"),

(8, "2023-02-15 12:00:00", "2023-02-25 12:00:00", "NFS Championship"),

(9, "2023-08-12 16:00:00", "2023-08-12 18:00:00", "Fast Food Tournamnet"),

(10, "2023-11-13 08:00:00", "2023-11-13 11:00:00", "Church"),

(11, "2023-02-05 9:00:00", "2023-02-10 23:00:00", "Super Bowl"),

(12, "2023-03-07 9:00:00", "2023-03-08 9:00:00", "75th Writers Guild of America Award"),

(14, "2023-05-01 20:00:00", "2023-05-02 3:00:00", "Lyrids Meteor Shower"),

(15, "2023-06-08 9:00:00", "2023-06-08 9:00:00", "Penumbral Lunar Eclipse "),

(16, "2023-07-09 9:00:00", "2023-07-09 9:00:00", "Supermoon"),

(17, "2023-08-10 22:00:00", "2023-08-11 3:00:00", "Blue Moon"),

(18, "2023-09-17 9:00:00", "2023-09-17 23:00:00", "Gran Prix Signapore"),

(19, "2023-11-17 9:00:00", "2023-11-17 19:00:00", "Melbourne Cup Horse Racing"),

(20, "2023-12-25 00:00:00", "2023-12-20 23:59:59", "Christmas");

INSERT INTO reservation VALUES

(1, 30, 'Dimitris', 'Peronis', 'ADULT'),

(2, 24, 'Pelopidas', 'Maximos', 'ADULT'),

(3, 31, 'Pavlos', 'Kratimenos', 'MINOR'),

(4, 27, 'Giorgia', 'Dekakoy', 'ADULT'),

(5, 39, 'Xristos', 'Koyrkoytis', 'MINOR'),

(6, 10, 'Sotiris', 'Gkekos', 'ADULT'),

(7, 4, 'Zina', 'Ziakoy', 'MINOR'),

(8, 34, 'Savvas', 'Loygkos', 'ADULT'),

(9, 20, 'Loykas', 'Arsipis', 'ADULT'),

(10, 58, 'Nikos', 'Sxisos', 'MINOR'),

(11, 1, "Holly", "Flax", "ADULT"),

(12, 1, "Kaitlyn", "Cole", "MINOR"),

(13, 6, "Agnes", "Knowles", "ADULT"),

(14, 3, "Autumn", "Murphy", "MINOR"),

(15, 7, "Isabel", "Todd", "ADULT"),

(16, 5, "Dominic", "Wallace", "MINOR"),

(17, 6, "Sonny", "Weaver", "ADULT"),

(18, 8, "Joe", "Gentry", "MINOR"),

(19, 2, "Ajay", "Benson", "ADULT"),

(20, 5, "Mathew", "Briggs", "MINOR");

INSERT INTO destination VALUES

(1, 'Paris', 'Beautiful', 'ABROAD', 'French', 'France'),

(2, 'Berlin', 'Majestic', 'ABROAD', 'German', 'Germany'),

(3, 'Stockholm', 'Lovely people', 'ABROAD', 'Swedish', 'Sweden'),

(4, 'Zagreb', 'Simple Life', 'ABROAD', 'Croatian', 'Croatia'),

(5, 'Mykonos', 'Gorgeous', 'LOCAL', 'Greek', 'Greece'),

(6, 'Santorini', 'Gorgeous', 'LOCAL', 'Greek', 'Greece'),

(7, 'Chios', 'Gorgeous', 'LOCAL', 'Greek', 'Greece'),

(8, 'Metsovo', 'Gorgeous', 'LOCAL', 'Greek', 'Greece'),

(9, 'Dublin', 'Whiskey', 'ABROAD', 'Irish', 'Ireland'),

(10, 'Dimitsana', 'Gorgeous', 'LOCAL', 'Greek', 'Greece'),

(11, "Hanover", "Beer", "ABROAD", "German", 'Germany'),

(12, "Frankfurt", "Croissant", "ABROAD", "French", 'France'),

(13, "Milan", "Passion", "ABROAD", "Italian", 'Italy'),

(14, "Kefalonia", "Beauty", "LOCAL", "Greek", 'Greece'),

(15, "Riyadh", "Rich", "ABROAD", "Arabic", 'Saudi Arabia'),

(16, "Osaka-shi", "Traditional", "ABROAD", "Japanese", 'Japan'),

(17, "Aarhus", "Architecture", "ABROAD", "Danish", 'Denmark'),

(18, "Krakow", "Auschwitz", "ABROAD", "Polish", 'Poland'),

(19, "Cambridge", "Rain", "ABROAD", "English", 'England'),

(20, "Seville", "Football", "ABROAD", "Spanish", 'Spain');

INSERT INTO travel\_to VALUES

(1, 1, "2022-09-03 16:30:00", "2022-09-09 15:00:00"),

(2, 2, "2022-07-12 17:00:00", "2022-07-12 22:00:00"),

(3, 3, "2022-11-15 11:00:00", "2022-11-17 13:00:00"),

(4, 4, "2022-09-01 09:00:00", "2022-09-08 20:00:00"),

(5, 5, "2022-02-02 12:00:00", "2022-02-05 03:00:00"),

(6, 6, "2022-01-14 14:00:00", "2022-01-14 08:00:00"),

(7, 7, "2023-05-22 18:00:00", "2023-05-22 22:00:00"),

(8, 8, "2023-02-15 14:00:00", "2023-02-25 10:00:00"),

(9, 9, "2023-08-12 19:00:00", "2023-08-12 14:00:00"),

(10, 10, "2023-11-13 10:00:00", "2023-11-13 09:00:00"),

(11, 11, "2023-02-03 10:00:00", "2023-02-09 16:00:00"),

(12, 12, "2023-03-07 14:00:00", "2023-03-11 19:00:00"),

(13, 13, "2023-04-02 15:00:00", "2023-04-08 10:00:00"),

(14, 14, "2023-05-01 16:00:00", "2023-05-06 12:00:00"),

(15, 15, "2023-06-08 10:00:00", "2023-06-13 10:00:00"),

(16, 16, "2023-07-09 12:00:00", "2023-07-16 02:00:00"),

(17, 17, "2023-08-10 10:00:00", "2023-08-19 13:00:00"),

(18, 18, "2023-09-12 13:00:00", "2023-09-20 06:00:00"),

(19, 19, "2023-11-17 17:00:00", "2023-11-23 18:00:00"),

(20, 20, "2023-12-20 15:00:00", "2023-12-30 02:00:00");

INSERT INTO it VALUES

("ATK94738", "hello", "2022-05-23", "2023-01-10");

INSERT INTO offers VALUES

(1, "2023-05-01", "2025-05-01", 500, 5),

(2, "2023-10-10", "2026-05-05", 700, 4),

(3, "2023-03-10", "2024-04-06", 450, 10);

## 1.4 Πολλαπλές εισαγωγές κρατήσεων προσφορών

Για την καταχώρηση πολλών εγγραφών στον πίνακα reservation\_offers, εκτελέσαμε τον εξής κώδικα:

SET GLOBAL local\_infile = 'ON';

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\names.csv' INTO TABLE reservation\_offers

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

(resof\_name, resof\_lname, @dummy, @dummy)

SET resof\_of\_id = 1,

resof\_paid = FLOOR(RAND() \* (200 - 50 + 1) + 50);

Μετατρέψαμε το αρχείο names.txt σε αρχείο csv, και στην συνέχεια με την εντολή LOAD DATA INFILE περάσαμε τα στοιχεία του ως τις δύο στήλες (όνομα και επίθετο) του πίνακα reservation\_offers. Οι δύο επόμενες στήλες με το “@” παραλείπονται και δεν γεμίζονται. Πρόκειται για τις στήλες resof\_of\_id, που δηλώσει την προσφορά στην οποία γίνεται η κράτηση και resof\_paid που δηλώνει το ποσό προκαταβολής που πλήρωσε το άτομο για να κάνει την κράτηση. Στο πεδίο του id της προσφοράς βάζουμε 1, ενώ στο πεδίο της προκαταβολής βάζουμε έναν τυχαίο αριθμό μεταξύ 250 και 500.

Με αυτόν τον κώδικα περνάμε 10.000 εγγραφές για την προσφορά 1. Τον εκτελέσαμε ακόμη 5 φορές, αλλάζοντας το πεδίο resof\_of\_id σε ένα από τα 2 και 3.

Έτσι, έχουμε 60.000 εγγραφές για τις 3 προσφορές μας.

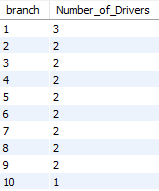
# Κεφάλαιο 2: Stored Procedures

## 2.1 new\_driver()

Το procedure παίρνει 7 ορίσματα: το AT του οδηγού (drv\_AT\_param), το όνομα του οδηγού (wrk\_name\_param), το επώνυμό του (wrk\_lname\_param), τον μισθό του (wrk\_salary\_param), την κατηγορία του διπλώματός του όπου επιλέγεται μία από τις προκαθορισμένες τιμές κατηγορίας διπλώματος A, B, C ή D (drv\_license\_param), το είδος των ταξιδιών που πραγματοποιεί, όπου πάλι επιλέγεται μια από τις προκαθορισμένες τιμές κατηγορίας ταξιδιών LOCAL ή ABROAD (drv\_route\_param) και την εμπειρία που κατέχει ως οδηγός (drv\_experience\_param).

Όταν καλέσουμε το procedure με τα στοιχεία του νέου μας οδηγού, θα βρει το υποκατάστημα που έχει τους λιγότερους οδηγούς, και θα τον κάνει INSERT στο συγκεκριμένο υποκατάστημα Αν έχουμε πολλά υποκαταστήματα που τυχαίνει να έχουν τον ίδιο ελάχιστο αριθμό οδηγών, τότε ο οδηγός θα μπαίνει σε ένα τυχαίο υποκατάστημα.

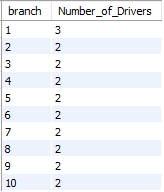
Έστω ότι έχουμε την παρακάτω κατάσταση, όπου το υποκατάστημα με br\_code = 10, έχει τους λιγότερους οδηγούς (number\_of\_drivers = 1):



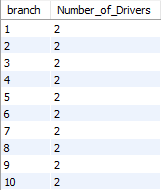
Αν καλέσουμε το procedure με τυχαία ορίσματα, έστω:



Παρατηρούμε ότι ο οδηγός μπήκε στο υποκατάστημα με br\_code = 10, που είχε τους λιγότερους οδηγού:



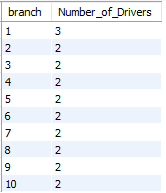
Έστω τώρα, ότι είχαμε την περίπτωση όπου όλα τα υποκαταστήματα τυγχάνει να έχουν τον ίδιο αριθμό οδηγών:



Αν καλέσουμε το ίδιο procedure:



Παρατηρούμε ότι ο οδηγός μπήκε σε ένα υποκατάστημα που επέλεξε τυχαία:



Τις παραπάνω εμφανίσεις πινάκων, τις εμφανίζουμε με τον εξής κώδικα:

SELECT br\_code AS branch, COUNT(\*) AS Number\_of\_Drivers

FROM worker

JOIN branch ON wrk\_br\_code = br\_code

JOIN driver ON wrk\_AT = drv\_AT

GROUP BY wrk\_br\_code

ORDER BY br\_code;

Ο κώδικας:

DROP PROCEDURE IF EXISTS new\_driver;

DELIMITER $$

CREATE PROCEDURE new\_driver(

IN drv\_AT\_param char(10),

IN wrk\_name\_param varchar(20),

IN wrk\_lname\_param varchar(20),

IN wrk\_salary\_param float(7,2),

IN drv\_license\_param enum('A','B','C','D'),

IN drv\_route\_param enum('LOCAL','ABROAD'),

IN drv\_experience\_param tinyint(4)

)

BEGIN

DECLARE bid int(11);

SELECT br\_code

INTO bid

FROM worker

JOIN branch ON wrk\_br\_code = br\_code

JOIN driver ON wrk\_AT = drv\_AT

GROUP BY br\_code

ORDER BY COUNT(drv\_AT)

LIMIT 1;

SELECT bid;

INSERT INTO worker VALUES(drv\_AT\_param, wrk\_name\_param, wrk\_lname\_param, wrk\_salary\_param, bid);

INSERT INTO driver VALUES(drv\_AT\_param, drv\_license\_param, drv\_route\_param, drv\_experience\_param);

END $$

DELIMITER ;

## 2.2 branch\_trip\_details()

To procedure παίρνει 3 ορίσματα: τον κωδικό του branch (brid), την πρώτη ημερομηνία (fdate) και μια δεύτερη ημερομηνία (sdate). Σαν αποτέλεσμα βγάζει για τα ταξίδια που διοργανώνει το συγκεκριμένο υποκατάστημα: κόστος ταξιδιού (tr\_cost ), μέγιστες θέσεις maxseat, σύνολο κρατήσεων (reservations), κενές θέσεις (maxseat – σύνολο κρατήσεων), επώνυμο και όνομα οδηγού και ξεναγού, ημερομηνία αναχώρησης και επιστροφής.

Παρακάτω βλέπουμε την κλήση της stored procedure branch\_trip\_details(), με την οποία ζητάμε τις πληροφορίες για κάθε ταξίδι που έχει οργανώσει το υποκατάστημα με id = 2 κατά την περίοδο ολόκληρου του έτους 2022:



Τα αποτελέσματα:



Πράγματι, το υποκατάστημα 2 έχει μόνο το συγκεκριμένο ταξίδι τις συγκεκριμένες μέρες.

Ο κώδικας:

DROP PROCEDURE IF EXISTS branch\_trip\_details;

DELIMITER $$

CREATE PROCEDURE branch\_trip\_details(IN brid INT, IN fdate DATE, IN sdate DATE)

BEGIN

DECLARE branch\_id INT;

DECLARE first\_date DATE;

DECLARE sec\_date DATE;

SET branch\_id = brid;

SET first\_date = fdate;

SET sec\_date = sdate;

SELECT tr\_id as trip\_id, tr\_cost as cost, tr\_maxseats as maxseats, COUNT(reservation.res\_tr\_id) as reservations, tr\_maxseats - COUNT(reservation.res\_tr\_id) as emptyseats, CONCAT(w1.wrk\_lname, ' ', w1.wrk\_name) as guide, CONCAT(w2.wrk\_lname, ' ', w2.wrk\_name) as driver, DATE(tr\_departure) as departure\_date, DATE(tr\_return) as return\_date

FROM trip JOIN worker as w1 ON w1.wrk\_AT = tr\_gui\_AT JOIN worker as w2 ON w2.wrk\_AT = tr\_drv\_AT JOIN reservation ON res\_tr\_id = tr\_id

WHERE tr\_br\_code = branch\_id AND DATE(tr\_departure) BETWEEN first\_date AND sec\_date

GROUP BY tr\_id;

END$$

DELIMITER ;

## 2.3 delete\_worker()

Το procedure παίρνει ως όρισμα τον ΑΤ του worker που θέλουμε να διαγράψουμε. Εάν αυτός είναι διευθυντής υποκαταστήματος εμφανίζει μήνυμα ότι δεν μπορεί να διαγραφθεί. Σε οποιαδήποτε άλλη περίπτωση διαγράφεται.

Καλούμε το procedure ως εξής:



Το αποτέλεσμα:



Πράγματι, ο υπάλληλος που πήγαμε να διαγράψουμε είναι διευθυντής του υποκαταστήματος 1 όπως φαίνεται παρακάτω:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Πάμε τώρα να διαγράψουμε έναν υπάλληλο, ο οποίος είναι διοικητικός, αλλά δεν είναι διευθυντής κάποιου υποκαταστήματος. Καλούμε την συνάρτηση με όρισμα τον ΑΤ του:



Και έχουμε το αποτέλεσμα:



Δηλαδή ο υπάλληλος διαγράφθηκε κανονικά.

Ο κώδικας:

DROP PROCEDURE IF EXISTS delete\_worker;

DELIMITER $$

CREATE PROCEDURE delete\_worker(IN workerid CHAR(10))

BEGIN

DECLARE worker\_id CHAR(10);

SET worker\_id = workerid;

IF EXISTS (SELECT \* FROM manages WHERE mng\_adm\_AT = worker\_id)

THEN

SELECT 'This worker is a branch manager and cannot be deleted.';

ELSE

DELETE FROM worker

WHERE wrk\_AT = worker\_id;

END IF;

END$$

DELIMITER ;

## 3.4

# Κεφάλαιο 3: Triggers

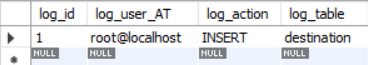
## 3.1 log updates

Ο πίνακας καταγραφής ενεργειών (log) ενημερώνεται κάθε φορά που εκτελείται μια αλλαγή (INSERT, UPDATE, DELETE) σε έναν από τους πίνακες trip, reservation, event, travel\_to, destination. Σε κάθε αλλαγή, αποθηκεύεται στον πίνακα log το id της, ο χρήστης που εκτέλεσε την αλλαγή, ο τύπος της αλλαγής και ο πίνακας.

Η εκτέλεση του κώδικα:



Επιφέρει την εξής αλλαγή στον πίνακα log:



Το παραπάνω παράδειγμα (και ο παρακάτω κώδικας) αντιστοιχεί το trigger της εισαγωγής στον πίνακα destination. Υπάρχουν triggers για όλους τους προαναφερόμενους πίνακες και για τις τρεις ενέργειες, αλλά εδώ παραθέτουμε μόνο έναν, καθώς οι άλλοι είναι όμοιοι.

Ο κώδικας:

DROP TRIGGER IF EXISTS desins;

DELIMITER $$

CREATE TRIGGER desins

AFTER INSERT

ON destination

FOR EACH ROW

BEGIN

INSERT INTO log VALUES(NULL, USER(), "INSERT", "destination");

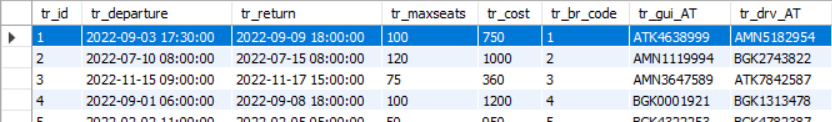
END$$

DELIMITER ;

## 3.2 update\_trip()

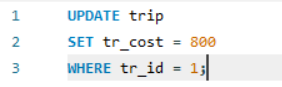
Με το trigger αυτό εμποδίζουμε τον χρήστη να αλλάξει τις ημερομηνίες αναχώρησης και επιστροφής, καθώς και το κόστος του ταξιδιού, εάν το ταξίδι αυτό έχει ήδη κρατήσης. Πριν κάθε ενημέρωση του πίνακα trip ελέγχεται εάν τα νέα στοιχεία που πρόκειται να καταχωρηθούν έχουν αλλαγές στις 3 στήλες που αναφέραμε. Εάν έχουν, το trigger δεν επιτρέπει την ενημέρωση και εκτυπώνεται στην οθόνη το αντίστοιχο signal.

Για παράδειγμα, εχουμε το ταξίδι 1, το οποίο έχει ήδη μια κράτηση:





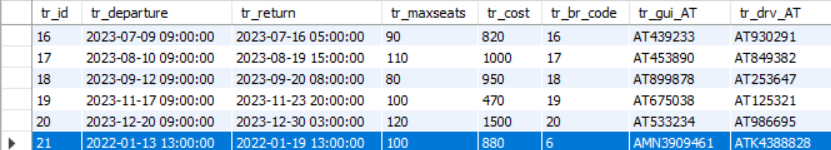
Εκτελώντας την ενημέρωση :



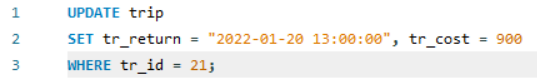
Προσπαθούμε να αλλάξουμε το κόστος του ταξιδιού από 750 σε 800. Όμως εμφανίζεται το εξής αποτέλεσμα:



Δηλαδή δεν μπόρεσε να γίνει η ενημέρωση. Εάν, τώρα, πάμε να κάνουμε ενημέρωση σε ένα ταξίδι, το οποίο δεν έχει κρατήσεις, η ενημέρωση θα γίνει κανονικά. Ας κάνουμε στο ταξίδι 21:



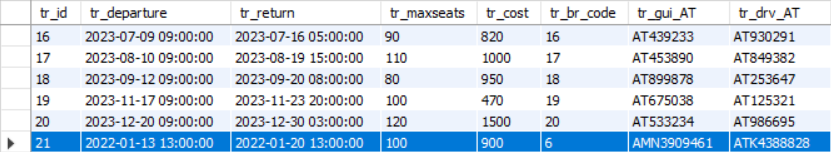
Εκτελούμε την ενημέρωση:



Και ως αποτέλεσμα έχουμε:



Και βλέπουμε ότι το ταξίδι έχει ενημερωθεί κατάλληλα:



Ο κώδικας:

DROP TRIGGER IF EXISTS update\_trip;

DELIMITER $$

CREATE TRIGGER update\_trip

BEFORE UPDATE

ON trip

FOR EACH ROW

BEGIN

IF (OLD.tr\_departure <> NEW.tr\_departure OR OLD.tr\_return <> NEW.tr\_return OR OLD.tr\_cost <> NEW.tr\_cost) AND

(SELECT COUNT(\*) FROM reservation WHERE res\_tr\_id = OLD.tr\_id) > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot update departure and arrival times or cost of trips with reservations';

END IF;

END$$

DELIMITER ;

## 3.3 changing\_salary()

Το trigger αυτό, αποτρέπει τον χρήστη από το να μειώσει τον μισθό κάποιου υπαλλήλου. Αναλυτικότερα, κάθε φορά που κάνουμε update σε έναν εργαζόμενο το πεδίο wrk\_salary, ελέγχεται με μια if η καινούργια τιμή που έβαλε ο χρήστης, με την παλιά τιμή του πεδίου wrk\_salary. Αν ο καινούργιος μισθός είναι μικρότερος από τον παλιό, εκτυπώνεται το SIGNAL “You can\’t set a lower salary”.

Πράγματι, στον πίνακα worker έχουμε την εξής εγγραφή.



Αν τρέξουμε την εντολή “UPDATE worker SET wrk\_salary=1000 WHERE wrk\_AT = ‘AMN0046398’;”, όπου wrk\_salary = 1000<2000, εκτυπώνεται στην οθόνη μας το Signal, ως error:



Αντίθετα, αν πάμε να προσθέσουμε μια τιμή στο πεδίο wrk\_salary μεγαλύτερη της προηγούμενης, η αλλαγή γίνεται κανονικά. Αν εκτελέσουμε την εντολή “UPDATE worker SET wrk\_salary=3000 WHERE wrk\_AT = 'AMN0046398';”, όπου wrk\_salary = 3000>2000, ο πίνακας αλλάζει ως εξής:



Ο κώδικας:

DROP TRIGGER IF EXISTS changing\_salary;

DELIMITER $$

CREATE TRIGGER changing\_salary

BEFORE UPDATE ON worker

FOR EACH ROW

BEGIN

IF (NEW.wrk\_salary < OLD.wrk\_salary)

THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT = 'You can\'t set a lower salary';

END IF;

END$$

DELIMITER ;

# Κεφάλαιο 4: GUI

## 4.1 Κώδικας Java για Insert

Ο κώδικάς μας όσον αφορά την εκτέλεση της λειτουργικότητας του πρώτου ζητουμένου για προσθήκη νέων εγγραφών στην βάση μας, είναι παρόμοιος για όλους τους πίνακες, με ελάχιστες διαφορές, όσον αφορά τα πεδία του κάθε πίνακα. Αναλυτικότερα, για όλους τους πίνακές μας, πρωταρχικός στόχος ήταν να μπορέσουμε να πάρουμε όλα τα ήδη υπάρχοντα δεδομένα από την βάση μας και να τα εμφανίσουμε στους πίνακες που δημιουργήσαμε στο GUI. Για να το καταφέρουμε αυτό, έπρεπε αρχικά να συνδεθούμε στην βάση μας με συγκεκριμένες μεθόδους όπως η getConnection(), για να μπορέσουμε να εκτελέσουμε την εντολή “SELECT \* FROM table” στην MySQL, που θα μας επιτρέψει να πάρουμε τα δεδομένα που χρειαζόμαστε. Αξίζει να σημειωθεί ότι όταν χρησιμοποιούμε την μέθοδο getConnection(), έπρεπε να βάζουμε κάθε φορά το username και το password που χρησιμοπούμε στην βάση της MySQL. Αυτό σημαίνει ότι για να τρέξει ο κώδικας με άλλη βάση δεδομένων, θα πρέπει να αλλαχθεί στον κώδικα το username και το password (οι μεταβλητές user και pass αντιστοίχως). Η εντολή της MySQL αποθηκεύεται σε ένα String και εκτελείται μέσω της μεθόδου executeQuery(). Τα αποτελέσματα της εντολής (δηλαδή όλες η εγγραφές του συγκεκριμένου πίνακα) αποθηκεύονται σε ένα ResultSet. Μέσω της δομής επανάληψης “while(myRs.next())”, διαπερνούμε μια μια τις αποθηκευμένες εγγραφές, τις τοποθετούμε προσωρινά σε ένα αντικείμενο που έχουμε ορίσει με όνομα το όνομα του πίνακα και ορίσματα τα ονόματα των πεδίων και τέλος αποθηκεύουμε το αντικείμενο αυτό στην λίστα που έχουμε φτιάξει στην αρχή του κώδικα. Το κομμάτι αυτό του κώδικα, αποτελεί σε κάθε αρχείο κώδικα την μέθοδο userList(). Στην συνέχεια, κατασκευάζουμε μια δεύτερη μέθοδο με όνομα show\_”table” (table = το όνομα του πίνακα που θέλουμε να φτιάξουμε κάθε φορά) στην οποία δημιουργούμε μια νέα λίστα list και την αρχικοποιούμε με την λίστα που επιστρέφει η μέθοδος userList() που χρησιμοποιήσαμε προηγουμένως. Δημιουργούμε επίσης ένα αντικείμενο με όνομα model το οποίο είναι τύπου DefaultTableModel και ως μορφή δεδομένων έχει το JTable που ορίσαμε στο GUI και ένα αντικείμενο row με αριθμό στοιχείων ίσο με τον αριθμό των στηλών στον πίνακα που θα γεμίσουμε. Αφού αποθηκεύσουμε με μια επανάληψη τις τιμές των στηλών του table τις οποίες τις παίρνουμε με αντίστοιχες μεθόδους που μας επιστρέφουν απλά το αποτέλεσμα του συγκεκριμένου πεδίου της κάθε στήλης, τα τοποθετούμε στον πίνακά μας με την εντολή “model.addRow(row);”. Καλώντας την μέθοδο show\_”table”, εμφανίζονται τα στοιχεία μας στον πίνακα. Αφού τελειώσουμε με την κατασκευή του πίνακα, συνέχεια έχει ο κουμπιού “Insert New Record” το οποίο εντάσσει την νέα εγγραφή μας στην βάση. Για να γίνει αυτό, αφού συνδεθούμε πάλι με την βάση μας, εκτελούμε αυτή την φορά την εντολή Insert της MySQL η οποία παίρνει με τις μεθόδους getText() ( για να πάρουμε τις εισαγωγές του χρήστη από τα πεδία του GUI) και setText() ( για να βάλουμε τις τιμές που πήραμε στη εντολή INSERT, στα VALUES(?,?,?,..)) τις τιμές που βάζει ο χρήστης. Για λόγους ασφαλείας και ελαχιστοποίησης λαθών η όλη διαδικασία γίνεται με PreparedStatement. Αφού βάλουμε στο VALUES(?,?,?,…) όλες τις τιμές που όρισε ο χρήστης, εκτελούμε την εντολή με την μέθοδο execute() που την καλή το PreparedStatement που ορίσαμε, εισάγοντας τα στοιχεία μας στην βάση. Έπειτα, προσθέτουμε τα στοιχεία της νέας εγγραφής στον πίνακα του GUI πάλι δημιουργώντας ένα αντικείμενο row με στήλες όσες στήλες του εκάστοτε πίνακα και ένα στιγμιότυπο model τύπου DeafultTableModel και αφού γίνουν και αυτές οι εισαγωγές, αδειάζουμε τα πεδία μας, ορίζοντάς τα με την “άδεια” τιμή “ ” για να μην χρειάζεται κάθε φορά ο χρήστης να σβήνει αυτά που έγραψε προηγουμένως.

Αξίζει επίσης να σημειωθεί ότι επίσης για λόγους ελαχιστοποίησης λαθών, σε πολλούς πίνακες όπου είχαμε σχέσεις ξένων κλειδιών και ο χρήστης έπρεπε να επιλέξει συγκεκριμένες τιμές τις βάσεις, χτίσαμε το πρόγραμμα έτσι ώστε να μην τον αφήνει να γράφει στα συγκεκριμένα πεδία, αλλά να επιλέγει την τιμή που θέλει. Ο κώδικας της συγκεκριμένης λειτουργικότητας είναι πολύ απλός αφού το μόνο που είχαμε να κάνουμε ήταν να συνδεθούμε πάλι στην βάση μας, να εκτελέσουμε την κατάλληλη εντολή SELECT για να πάρουμε τις τιμές που θέλαμε, να τις αποθηκεύσουμε σε ένα ResultSet και να τις βάλουμε στο combo box του GUI.

Σε πολλές περιπτώσεις, είχε σημασία και η σειρά με την οποία βάζαμε τα δεδομένα στην βάση μας. Αναλυτικότερα, όπως έχουμε ήδη προαναφέρει, πολλοί πίνακες είχαν σχέσεις μεταξύ τους με ξένα κλειδιά. Αυτό σημαίνει ότι άμα για παράδειγμα θέλαμε να ορίσουμε έναν καινούργιο driver, θα έπρεπε πρώτα να αποθηκεύσουμε τα απαραίτητα στοιχεία στον πίνακα worker και έπειτα να βάλουμε τα απαραίτητα στοιχεία στον πίνακα driver. Αυτό όσον αφορά τον κώδικά μας, σημαίνει ότι έπρεπε πρώτα να εκτελέσουμε μια εντολή INSERT με τα ανάλογα δεδομένα στον πίνακα worker και μετά μία εντολή INSERT για τον πίνακα driver, έτσι ώστε να μην προκύπτει error στην MySQL και Exception στην Java. Η δομή με την οποία έχει κατασκευαστεί το πρόγραμμά μας αποτρέπει τον χρήστη από το να κάνει τέτοια λάθη.

## 4.2 Κώδικας Java για ζητούμενο εισαγωγής νέου υπεύθυνου πληροφορικής με ίσα δικαιώματα

Ο κώδικάς μας, είναι παρόμοιος με αυτόν που περιγράφεται στο “Κώδικας Java για Insert”. Όσον αφορά τα δικαιώματα, αφού στο παράθυρο Login παίρνουμε το username και το password που βάζει ο χρήστης και τα ελέγχουμε με τα αναγκαία επίπεδα από τον πίνακα IT, σημαίνει ότι στο πρόγραμμα θα μπορούν να μπουν μόνο όσοι είναι καταγεγραμμένοι στον πίνακα IT. Άρα, θα έχουν ίδια δικαιώματα.

## 4.3 Κώδικας Java για ζητούμενο σχετικά με πληροφορίες υποκαταστήματος, διευθυντή υποκαταστήματος, σύνολο κρατήσεων και σύνολο εσόδων

Στο αρχείο Branch\_Profits\_Details.java, το οποίο είναι υπεύθυνο για το ζητούμενο αυτό, έπρεπε να ασχοληθούμε δύο κομμάτια κώδικα. Πρώτον με το κουμπί “Show Branch Codes” το οποίο όπως έχουμε προαναφέρει κάνει connection με την βάση μας, εκτελεί την εντολή “SELECT br\_code FROM branch”, αποθηκεύει τα αποτελέσματα σε ένα ResultSet και τα τοποθετεί εν τέλει στο combo box που εμφανίζεται στην οθόνη μας. Ανάλογα με τον κωδικό υποκαταστήματος που επιλέγει ο χρήστης, θα πρέπει με το πάτημα του κουμπιού “Show Data” να εμφανίζονται στα πεδία μας όλες οι ζητούμενες τιμές. Όσον αφορά το κομμάτι του κώδικα, εκτελέσαμε διαδοχικά τρεια String. Το “String sql1 = "SELECT br\_street, br\_num, br\_city FROM branch WHERE br\_code = " + brc;” στο οποίο παίρνουμε πληροφορίες για το υποκατάστημα που επέλεξε ο χρήστης. Το επιλεγμένο branch code το επιλέξαμε με την μέθοδο getSelectedItem() και στην συνέχεια το αποθηκεύσαμε σε μια μεταβλητή με όνομα brc, αφού πρώτα το μετατρέψαμε σε ακέραιο με την εντολή “brc = Integer.parseInt(brc\_str);” για να αναγνωρίζει τον ακέραιο η MySQL. Με την μέθοδο setText γεμίσαμε τα αντίστοιχα πεδία με τις τιμές που θέλαμε, τις οποίες τις πήραμε από το Result Set με την μέθοδο getString() (στο ResultSet αποθηκεύουμε κάθε φορά γραμμή γραμμή τα περιεχόμενα του πίνακα που χρησιμοποιούμε κάθε φορά). Αντιστοίχως, εκτελέσαμε το String “String sql2 = "SELECT wrk\_name, wrk\_lname FROM worker JOIN branch ON wrk\_br\_code = br\_code JOIN manages ON mng\_adm\_AT = wrk\_AT WHERE br\_code = " +brc;” για να πάρουμε το όνομα και το επίθετο αυτού που διευθύνει το υποκατάστημα και τέλος εκτελέσαμε το String “String sql3 = "SELECT COUNT(\*), SUM(tr\_cost) FROM trip JOIN branch ON tr\_br\_code = br\_code WHERE br\_code =? GROUP BY tr\_br\_code";” για το σύνολο κρατήσεων και το σύνολο εσόδων. Για το σύνολο κρατήσεων κάναμε GROUP BY tr\_br\_code αφού για κάθε trip αντιστοιχεί ένα reservation, οπότε άμα τα ομαδοποιήσουμε με βάση τον κωδικό του υποκαταστήματος και κάνουμε COUNT(\*) θα έχουμε τον αριθμό των κρατήσεων. Τέλος, εκτελέσαμε το String “String sql4 = "SELECT SUM(resof\_paid) FROM offers JOIN reservation\_offers ON resof\_of\_id=of\_id WHERE of\_br\_code="+brc;” μέσω του οποίου παίρνουμε το άθροισμα όλων των προκαταβολών μας (επειδή είχαμε βάλει τις 60000 reservation\_offers, τα αποτελέσματα που προκύπτουν είναι αρκετά μεγάλα). Οπότε, για το σύνολο των εσόδων μας, αθροίζουμε όλες τις προκαταβολές, μαζί με τα έσοδα των ταξιδιών.

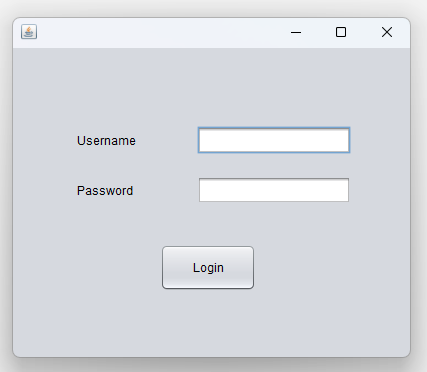
## 4.4 Κώδικας Java για ζητούμενο σχετικά ονοματεπώνυμα υπαλλήλων, και πληροφορίες μισθών

Στο αρχείο Branch\_Pays\_Details.java, το οποίο είναι υπεύθυνο για το ζητούμενο αυτό, έπρεπε να ασχοληθούμε τρία κομμάτια κώδικα. Πρώτον με το κουμπί “Show Branch Codes” το οποίο όπως έχουμε προαναφέρει κάνει connection με την βάση μας, εκτελεί την εντολή “SELECT br\_code FROM branch”, αποθηκεύει τα αποτελέσματα σε ένα ResultSet και τα τοποθετεί εν τέλει στο combo box που εμφανίζεται στην οθόνη μας. Δεύτερον, με το κουμπί “Show Total” το οποίο αφού εκτελέσει το String “String sql = "SELECT wrk\_salary FROM worker JOIN branch ON wrk\_br\_code = br\_code WHERE br\_code =" + brc;” για να πάρει τους μισθούς των υπαλλήλων και να τους αθροίσει μέσα σε μια while(myRs.next()) (οι μισθοί είναι αποθηκευμένοι σε ένα ResultSet με όνομα myRs), πρέπει να εκτελέσει και το String “String sql2 = "SELECT SUM(wrk\_salary) FROM it JOIN worker ON it\_AT=wrk\_AT";” για να πάρει και το άθροισμα των μισθών των υπεύθυνων του IT οι οποίοι είναι στον ξεχωριστό πίνακα it και δουλεύουν για όλα τα υποκαταστήματα. Αφού αθροίσουμε όλους τους μισθούς, παίρνουμε το αποτέλεσμα των συνολικών εξόδων. Τέλος, με το κουμπί “Refresh Data” καλούμε την μέθοδο “userList” που περιγράφαμε στο κομμάτι (4.1 Κώδικας Java για Insert) για να γεμίσουμε την λίστα μας με τα δεδομένα που χρειαζόμαστε. Έπειτα, καλούμε την μέθοδο “remove\_data\_table()” μέσω της οποίας διαγράφουμε τα στοιχεία που είχε προηγουμένως ο πίνακας του GUI μας και καλούμε την μέθοδο “show\_branch\_pays()” που εμφανίζει τα ζητούμενα δεδομένα που αποθηκεύσαμε στην λίστα, στον πίνακα του GUI (η μέθοδος περιγράφεται και στο 4.1 Κώδικας Java για Insert αναλυτικότερα).

Αξίζει να σημειωθεί ότι αντιμετωπίσαμε ένα θέμα όσον αφορά το combo box. Μέχρι να πατήσουμε το “Show Branch Codes” το combo box είχε την τιμή NULL που σημαίνει ότι ΄΄οταν προσπαθούσαμε να κάνουμε Build το πρόγραμμα, μας έβγαζε exception γιατί προφανώς δεν έβρισκε κωδικό υποκαταστήματος στην βάση μας με τέτοια τιμή, ώστε να μας εμφανίσει τα ζητούμενα δεδομένα. Για την αντιμετώπισή του (ίσως όχι βέλτιστη), αρχικοποιήσαμε το combo box με την τιμή 1 έτσι ώστε όταν γίνεται build να έχει μια υπάρχουσα τιμή να αντλήσει δεδομένα και αφού πατηθεί το κουμπί “Show Branch Codes” εμφανίζονται και όλοι οι υπόλοιποι κωδικοί.

## 4.5 Σύνδεση

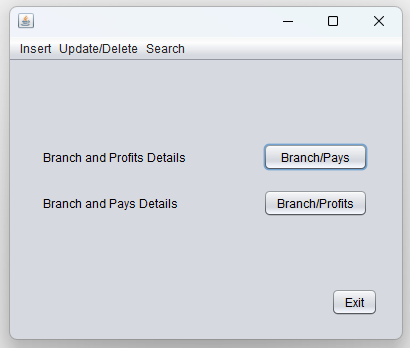
Ξεκινώντας την εκτέλεση του κώδικα, μας εμφανίζεται το login παράθυρο όπου οι χρήστες με δικαιώματα IT μπορούν να βάλουν τα στοιχεία τους και να συνδεθούν στο πρόγραμμα.



Ως IT έχουμε ορίσει την εξής εγγραφή:



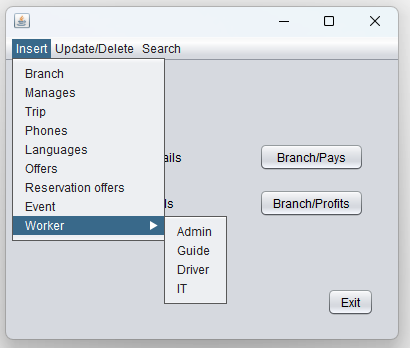
Βάζοντας τα στοιχεία αυτά στο Username και στο Password, εισερχόμαστε στο menu του προγράμματος, όπου εκτελούνται και όλες οι ζητούμενες ενέργειες.



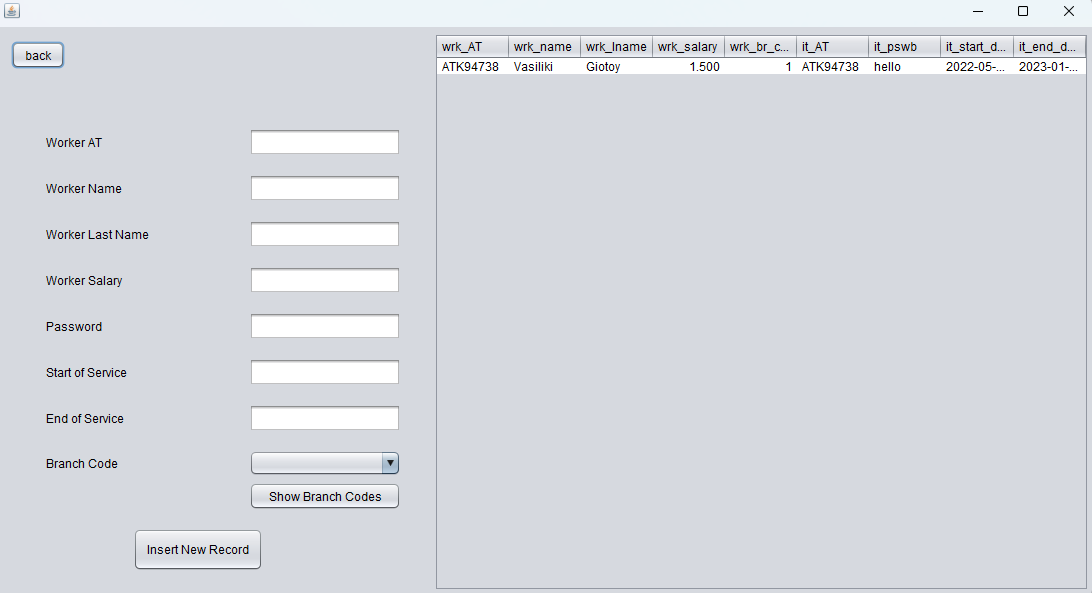
## 4.6 Εισαγωγή Εγγραφής

Όσον αφορά το πρώτο ζητούμενο με τις εισαγωγές, έχουμε την καρτέλα “Insert” όπου μπορούν να γίνουν όλες οι εισαγωγές. Πολλοί πίνακες επειδή σχετίζονται μεταξύ τους με σχέσεις ξένων κλειδιών, όπως για παράδειγμα ο πίνακας worker με τους admin, driver, it και guide, ή όπως ο πίνακας trip με τους reservation, destination και trave\_to κτλπ, έπρεπε να φτιαχτούν με τέτοιο τρόπο έτσι ώστε πρώτα να γίνεται η εγγραφή στον πίνακα που αποτελεί την ισχυρή οντότητα, και ύστερα στους πίνακες που αποτελούν τις ασθενείς οντότητες.

Έστω για παράδειγμα, ότι πάμε να κάνουμε μια νέα εγγραφή στον πίνακα it:



Αφού επιλέξουμε το IT, μας ανοίγει το παράθυρο για να κάνουμε νέες εγγραφές:



Στο παράθυρο αυτό, έχουμε έναν πίνακα που μας δείχνει κάθε φορά τα δεδομένα του συγκεκριμένου πίνακα στην βάση μας. Περιέχει πεδία που τα συμπληρώνουμε με τις τιμές που θέλουμε να βάλουμε για την νέα εγγραφή μας και ένα κουμπί με όνομα “Show Branch Codes” το οποίο όταν το πατήσουμε, γεμίζει το πεδίο Branch Code με όλα τα δυνατά id υποκαταστημάτων που μπορεί να επιλέξει ο χρήστης για να εργαστεί ο νέος IT υπάλληλος. Τέλος, έχουμε φτιάξει και ένα κουμπί back το οποίο όταν πατηθεί μας πηγαίνει στο προηγούμενο παράθυρο, δηλαδή στην προκειμένη περίπτωση το μενού.

Έστω, ότι θέλουμε να κάνουμε μια νέα εγγραφή με τα εξής δεδομένα:

Worker AT = ATK153674

Worker Name = Giorgos

Worker Last Name = Vasilioy

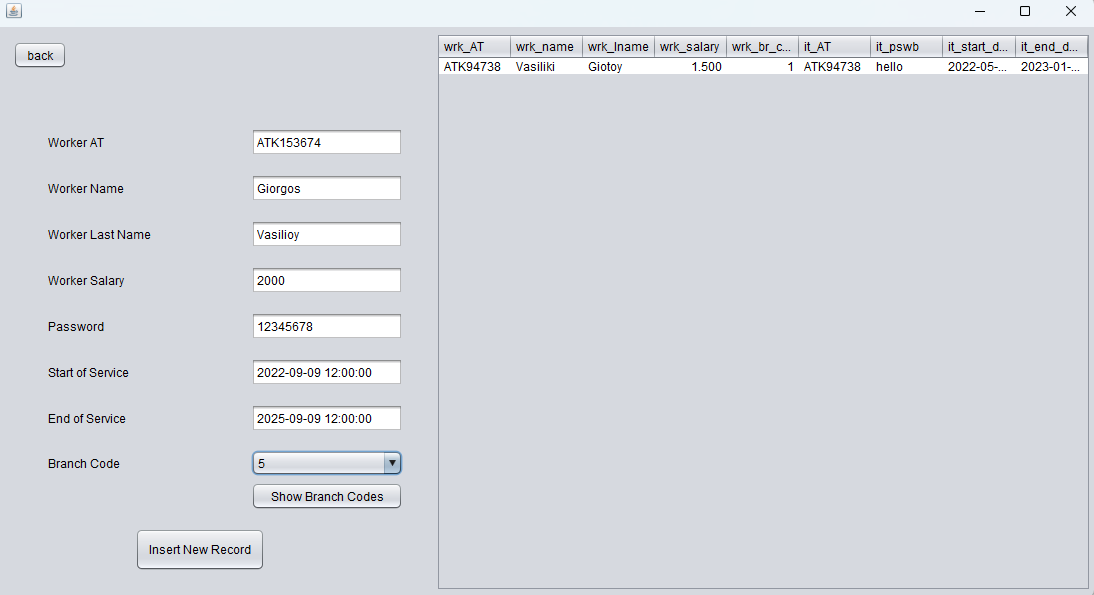
Worker Salary = 2000

Password = 12345678

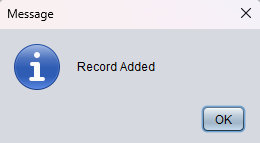
Start of Service = 2022-09-09 12:00:00

End of Service = 2025-09-09 12:00:00

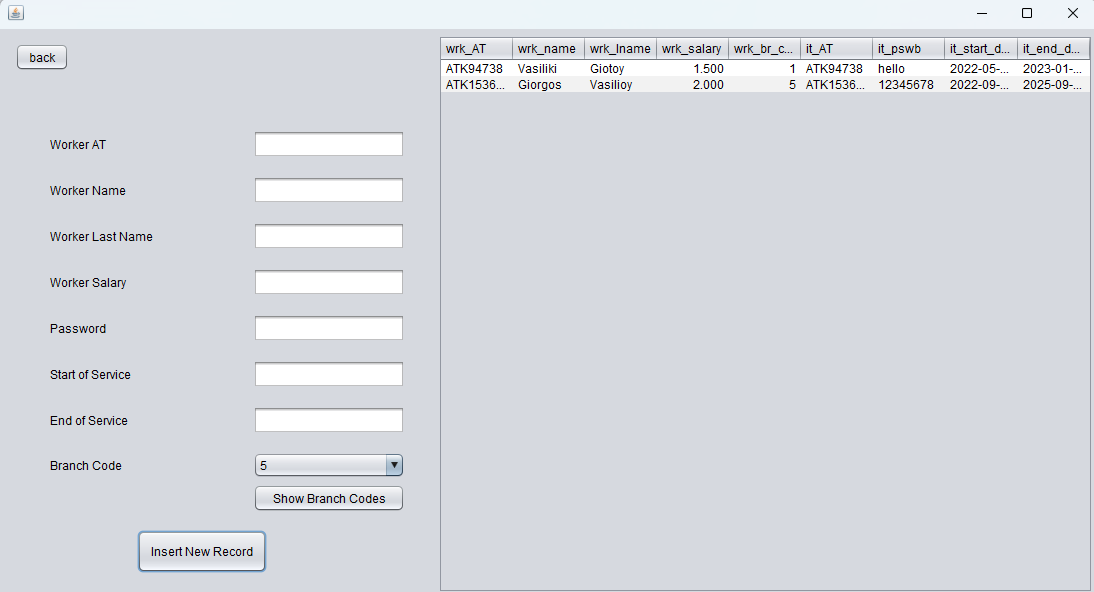
Branch Code = 5

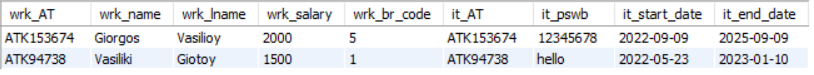


Αν πατήσουμε το κουμπί “Insert New Record” μέσω του οποίου εισάγουμε την νέα εγγραφή μας στην βάση δεδομένων, θα μας εκτυπωθεί το εξής μήνυμα επιτυχίας εισαγωγής:



Επίσης, παρατηρούμε ότι η νέα εγγραφή μας φαίνεται και στον πίνακά μας με τις εγγραφές:

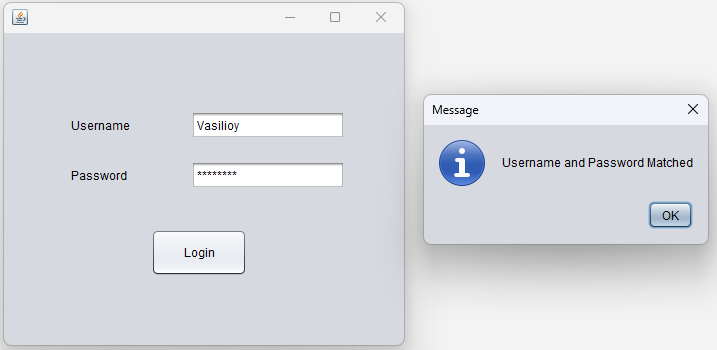
Να σημειώσουμε ότι αν πάμε στην mySQL για να δούμε αν όντως έχει γίνει η εισαγωγή στην βάση δεδομένων, θα παρατηρήσουμε ότι η εγγραφή εκτελέστηκε κανονικά:

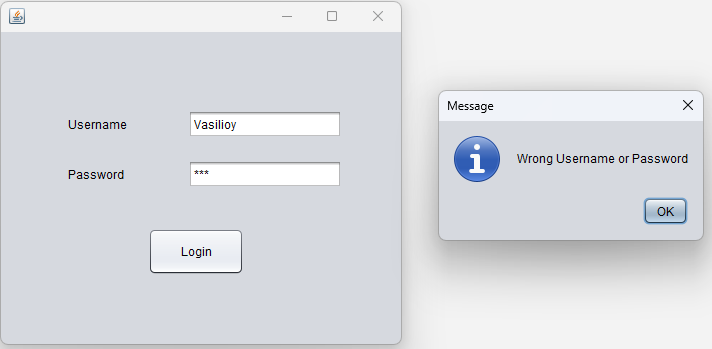


Με παρόμοιο τρόπο λειτουργούν όλες οι εισαγωγές δεδομένων στην βάση μας.

### 4.7 Δικαιώματα Νέου IT Υπαλλήλου

Στο σημείο αυτό, αξίζει να τονίσουμε ότι αφού η εγγραφή μας είναι ένας νέος υπάλληλος στο IT, σημαίνει ότι θα πρέπει να αποκτήσει τα ίδια δικαιώματα με τους υπόλοιπους υπαλλήλους του IT. Δηλαδή, να μπορεί να προσθέσει νέες εγγραφές, να διαγράψει και να αλλάξει τις ήδη υπάρχουσες κτλπ. Επίσης, σημαίνει ότι άμα χρησιμοποιήσουμε το επώνυμό του και τον κωδικό του, θα μπορέσουμε να συνδεθούμε στο πρόγραμμα. Αναλυτικότερα, βάζοντας τα στοιχεία του στο Login παράθυρο, έχουμε:

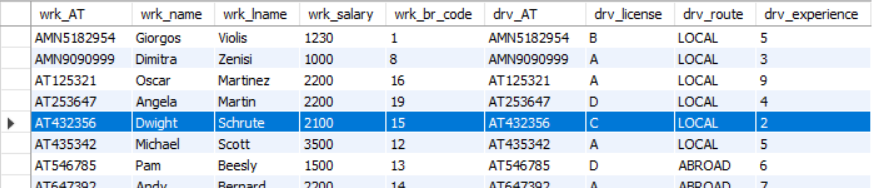
Αν βάζαμε λάθος Username ή Password, προφανώς δεν θα μπορούσαμε να συνδεθούμε. Για παράδειγμα έστω ότι βάζουμε λάθος κωδικό με λιγότερα ψηφία από τον πραγματικό (για να φαίνεται η διαφορά):



## 4.8 Ενημέρωση Εγγραφής

Ας υποθέσουμε ότι θέλουμε να ενημερώσουμε τα στοιχεία ενός οδηγού (πίνακας driver). Ο οδηγός έχει AT = “AT432356”. Ελέγχουμε τα στοιχεία του στην βάση μας πριν εκτελέσουμε την ενημέρωση:

(Εμφανίζουμε όλα τα στοιχεία του εργαζομένου, τόσο από τον πίνακα driver όσο και από τον πίνακα worker, καθώς η ενημέρωση μπορεί να γίνει σε όλα)



Στο μενού επιλέγουμε με την σειρά τις καρτέλες Update/Delete -> Worker -> Driver

Πατώντας το κουμπί **DRIVER** εμφανίζεται το εξής παράθυρο:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Για να εμφανίσουμε τον πίνακα πατάμε το κουμπί **VIEW** (αριστερή στήλη).

Με το που πατήσουμε VIEW εκτελούνται οι εξής εντολές:

st = myConn.createStatement();

String sql = "SELECT \* FROM worker JOIN driver ON wrk\_AT = drv\_AT";

myRs = st.executeQuery(sql);

Δηλαδή δημιουργείται η σύνδεση με την βάση δεδομένων μας και επιλέγουμε όλα τα στοιχεία των οδηγών. Τα στοιχεία αυτά τα προβάλλουμε στον πίνακα ως εξής:

while (myRs.next()) {

String wrk\_AT = String.valueOf(myRs.getString("wrk\_AT"));

String wrk\_name = String.valueOf(myRs.getString("wrk\_name"));

String wrk\_lname = String.valueOf(myRs.getString("wrk\_lname"));

String wrk\_salary = String.valueOf(myRs.getInt("wrk\_salary"));

String wrk\_br\_code = String.valueOf(myRs.getInt("wrk\_br\_code"));

String drv\_license = String.valueOf(myRs.getString("drv\_license"));

String drv\_route = String.valueOf(myRs.getString("drv\_route"));

String drv\_experience = String.valueOf(myRs.getString("drv\_experience"));

String tbData[] = {wrk\_AT, wrk\_name, wrk\_lname, wrk\_salary, wrk\_br\_code, drv\_license, drv\_route, drv\_experience};

tb1Model.addRow(tbData);

}

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Βλέπουμε ότι δεξιά εμφανίζονται ο πίνακας με τα στοιχεία όλων των οδηγών.

Εντοπίζουμε με το ποντίκι μας τον εργαζόμενο που θέλουμε:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Με το που πατήσουμε συγκεκριμένη εγγραφή, τα στοιχεία κάθε στήλης μεταφέρονται στα πλαίσια κειμένου, τα οποία μπορούμε να επεξεργαστούμε.

Αλλάζουμε τον μισθό (salary) σε 2500 και την εμπειρία (experience) σε 3:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Μόλις είμαστε έτοιμοι και αλλάξαμε τα στοιχεία που θέλαμε, πατάμε **UPDATE** (αριστερή στήλη). ΑΝ δεν προκύψει κάποιο πρόβλημα, εμφανίζεται μήνυμα για επιτυχή ενημέρωση.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Οι εντολές για την ενημέρωση είναι:

String sql = "UPDATE driver JOIN worker ON wrk\_AT = drv\_AT SET wrk\_name='" + uname + "', wrk\_lname='" + ulname + "', wrk\_salary='" + usalary + "', wrk\_br\_code='" + ubranch + "', drv\_license='" + ulicense + "', drv\_route='" + uroute + "', drv\_experience='" + uexp + "' WHERE drv\_AT='" + uat + "';";

st = myConn.prepareStatement(sql);

st.executeUpdate(sql);

Μόλις πατάμε να γίνει η ενημέρωση, ο πίνακας φορτώνεται ξανά και μπορούμε να δούμε ότι οι αλλαγές έχουν περαστεί.

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Ας δούμε τώρα τον πίνακα από την βάση μας:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Παρατηρούμε πως οι αλλαγές έχουν περαστεί κανονικά στην βάση μας, άρα η ενημέρωση ήταν επιτυχής.

## 4.9 Διαγραφή Εγγραφής

Για να εκτελέσουμε ενέργεια διαγραφής, επιλέγουμε από πάνω την καρτέλα Update/Delete και στην συνέχεια τον πίνακα από τον οποίο θέλουμε να διαγράψουμε μια εγγραφή. Ας υποθέσουμε ότι θέλουμε να διαγράψουμε τον ξεναγό με ΑΤ = “AT453890”.

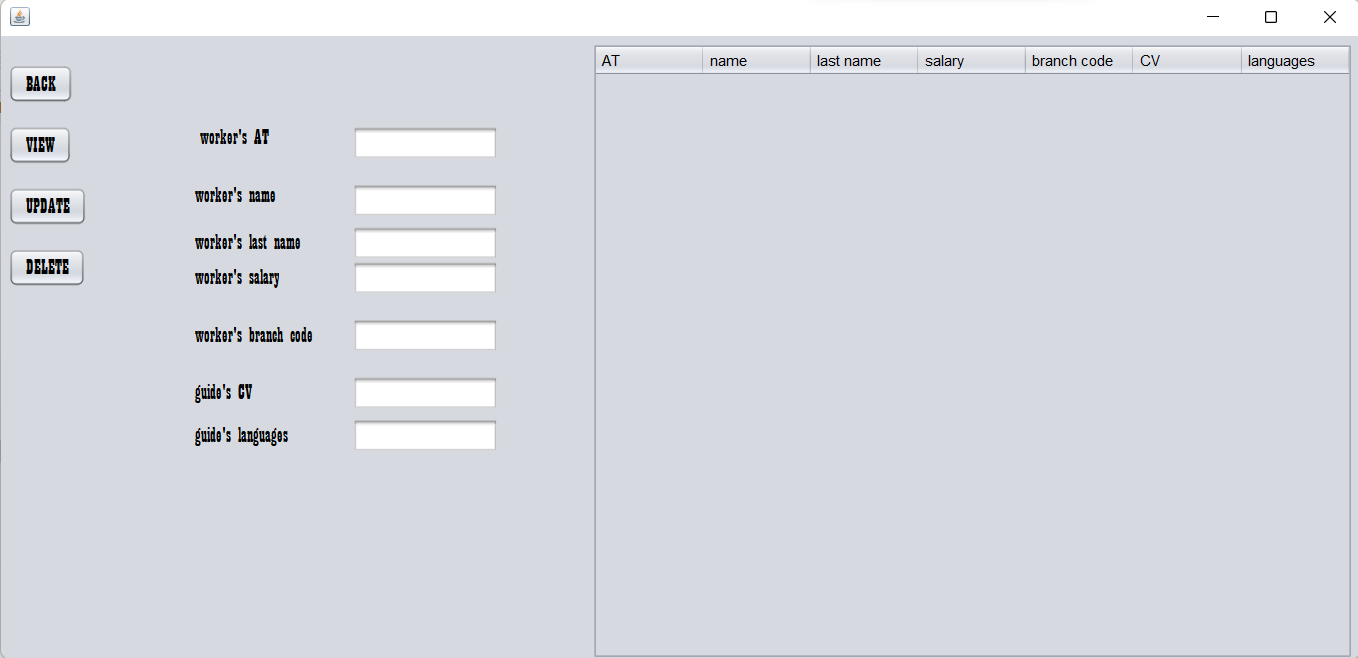
Παρακάτω βλέπουμε ότι ο οδηγός αυτός βρίσκεται όντως στην βάση μας:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Επιλέγουμε από το μενού με την σειρά τις καρτέλες Update/Delete -> Worker -> Guide

Πατώντας το κουμπί **GUIDE** εμφανίζεται το εξής παράθυρο:



Για να εμφανίσουμε τον πίνακα πατάμε το κουμπί **VIEW** (αριστερή στήλη).

Μόλις πατήσουμε το κουμπί VIEW, εκτελείται ο εξής κώδικας:

myConn = DriverManager.getConnection("jdbc:mysql://localhost:3306/travel\_agency", user, pass);

st = myConn.createStatement();

String sql = "SELECT \* FROM worker JOIN guide ON wrk\_AT = gui\_AT JOIN languages ON lng\_gui\_AT = gui\_AT";

myRs = st.executeQuery(sql);

DefaultTableModel tb1Model = (DefaultTableModel) jTableGuide.getModel();

tb1Model.setRowCount(0);

while (myRs.next()) {

String wrk\_AT = String.valueOf(myRs.getString("wrk\_AT"));

String wrk\_name = String.valueOf(myRs.getString("wrk\_name"));

String wrk\_lname = String.valueOf(myRs.getString("wrk\_lname"));

String wrk\_salary = String.valueOf(myRs.getInt("wrk\_salary"));

String wrk\_br\_code = String.valueOf(myRs.getInt("wrk\_br\_code"));

String gui\_cv = String.valueOf(myRs.getString("gui\_cv"));

String gui\_lang = String.valueOf(myRs.getString("lng\_language"));

String tbData[] = {wrk\_AT, wrk\_name, wrk\_lname, wrk\_salary, wrk\_br\_code, gui\_cv, gui\_lang };

tb1Model.addRow(tbData);

}

Με τον κώδικα αυτό, γίνεται η σύνδεση με την βάση δεδομένων μας και στην συνέχεια εκτελείται το query (SELECT \* FROM WORKER JOIN GUIDE ON wrk\_AT = gui\_AT). Έτσι επιλέγουμε για εμφάνιση όλα τα στοιχεία των οδηγών, μαζί με αυτά που είναι αποθηκευμένα στον πίνακα worker. Έπειτα, τα περνάμε στον πίνακα που έχουμε ετοιμάσει στα δεξιά.

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Βλέπουμε ότι δεξιά εμφανίζονται ο πίνακας με τα στοιχεία όλων των ξεναγών.

Εντοπίζουμε με το ποντίκι μας τον εργαζόμενο που θέλουμε να διαγράψουμε:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Αφού τον έχουμε επιλέξει, πατάμε το κουμπί **DELETE** (αριστερή στήλη). Έτσι εκτελούνται οι εξής εντολές:

if (jTableGuide.getSelectedRowCount() == 1) {

String uat = aat.getText();

String uname = aname.getText();

String ulname = alname.getText();

String usalary = asalary.getText();

String ubranch = abranch.getText();

String ucv = dcv.getText();

String ulang = guilanguages.getText();

myConn = DriverManager.getConnection("jdbc:mysql://localhost:3306/travel\_agency", user, pass);

String sql = "DELETE FROM worker WHERE wrk\_AT='" + uat + "';";

st = myConn.prepareStatement(sql);

st.executeUpdate(sql);

Αφού δημιουργήσουμε για ακόμη μια φορά την σύνδεση με την βάση δεδομένων μας, εκτελούμε το query για την διαγραφή του ξεναγού από τον πίνακ worker. Εδώ να σημειώσουμε πως εφόσον οι πίνακες worker και guide συσχετίζονται, με την διαγραφή του ξεναγού από τον πίνακα worker διαγράφεται και από το πίνακα guide.

Αν η διαγραφή είναι επιτυχής, εμφανίζεται το μήνυμα:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Μόλις κλείσουμε το μήνυμα, ο πίνακας φορτώνεται πάλι και βλέπουμε ότι ο ξεναγός έχει όντως διαγραφθεί.

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Ας δούμε τώρα τον πίνακα από την βάση μας:

Εικόνα που περιέχει πίνακας

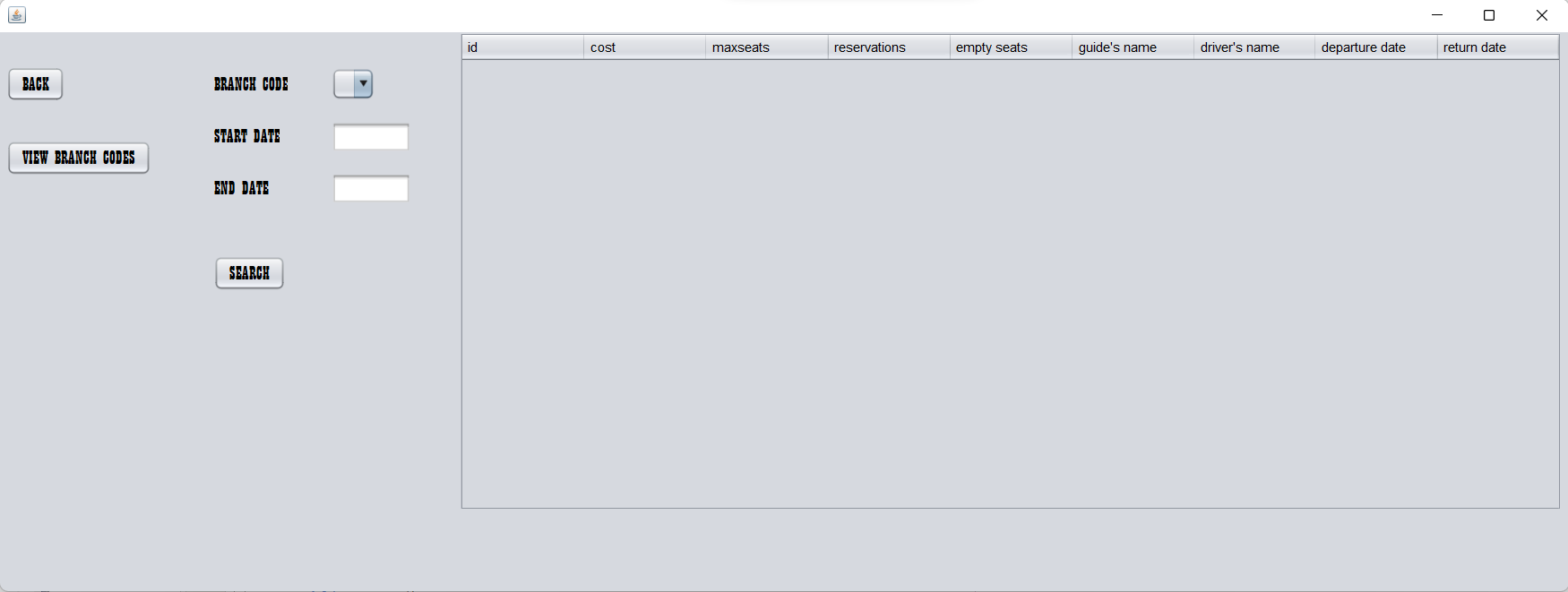
Περιγραφή που δημιουργήθηκε αυτόματα

Ο ξεναγός πλέον δεν υπάρχει, ούτε στον πίνακα worker, ούτε στον πίνακα guide. Η διαγραφή έγινε επιτυχώς.

## 4.10 Ζητούμενο για προβολή λεπτομερειών ταξιδιού

Όπως περιγράψαμε και στο κεφάλαιο 2.2, έχουμε υλοποιήσει ένα procedure, το οποίο παίρνει το id ενός υποκαταστήματος, μαζί με δύο ημερομηνίες και επιστρέφει λεπτομέρειες για όλα τα ταξίδια του που θα πραγματοποιηθούν στο διάστημα αυτό.

Επιλέγουμε από το μενού με την σειρά τις καρτέλες search -> branch -> by dates και εμφανίζεται το εξής παράθυρο:



Πατώντας το κουμπί VIEW BRANCH CODES φορτώνονται όλα τα id των υποκαταστημάτων στο μενού branch code, από το οποίο επιλέγουμε το υποκατάστημα στο οποίο θέλουμε να κάνουμε την αναζήτηση. Με το που επιλέγουμε να εμφανιστούν τα ids, γίνεται σύνδεση με την βάση δεδομένων μας και εκτελούνται οι εξής εντολές, με τις οποίες προστίθεται οι κωδικοί των υποκαταστημάτων στην μπάρα επιλογής:

st = myConn.createStatement();

String sql = "SELECT br\_code FROM branch";

myRs = st.executeQuery(sql);

while (myRs.next()) {

String id = myRs.getString("br\_code");

branch.addItem(id);

}

Έπειτα βάζουμε και τις δύο ημερομηνίες, και πατάμε το κουμπί SEARCH.

Μόλις πατήσουμε να γίνει η αναζήτηση, δημιουργείται νέα σύνδεση με την βάση δεδομένων μας και καλείται το procedure που υλοποιήσαμε, όπου selectedValue είναι η επιλεγμένη τιμή του branch id, start η πρώτη ημερομηνία και end η δεύτερη ημερομηνία. :

st = myConn.createStatement();

String sql = "CALL branch\_trip\_details( '" + selectedValue + "', '" + start + "', '" + end + "' )";

myRs = st.executeQuery(sql);

Μετά την σύνδεση φορτώνουμε τα αποτελέσματα του procedure στον πίνακά μας:

while (myRs.next()) {

String c1 = String.valueOf(myRs.getString("trip\_id"));

String c2 = String.valueOf(myRs.getString("cost"));

String c3 = String.valueOf(myRs.getString("maxseats"));

String c4 = String.valueOf(myRs.getString("reservations"));

String c5 = String.valueOf(myRs.getString("emptyseats"));

String c6 = String.valueOf(myRs.getString("guide"));

String c7 = String.valueOf(myRs.getString("driver"));

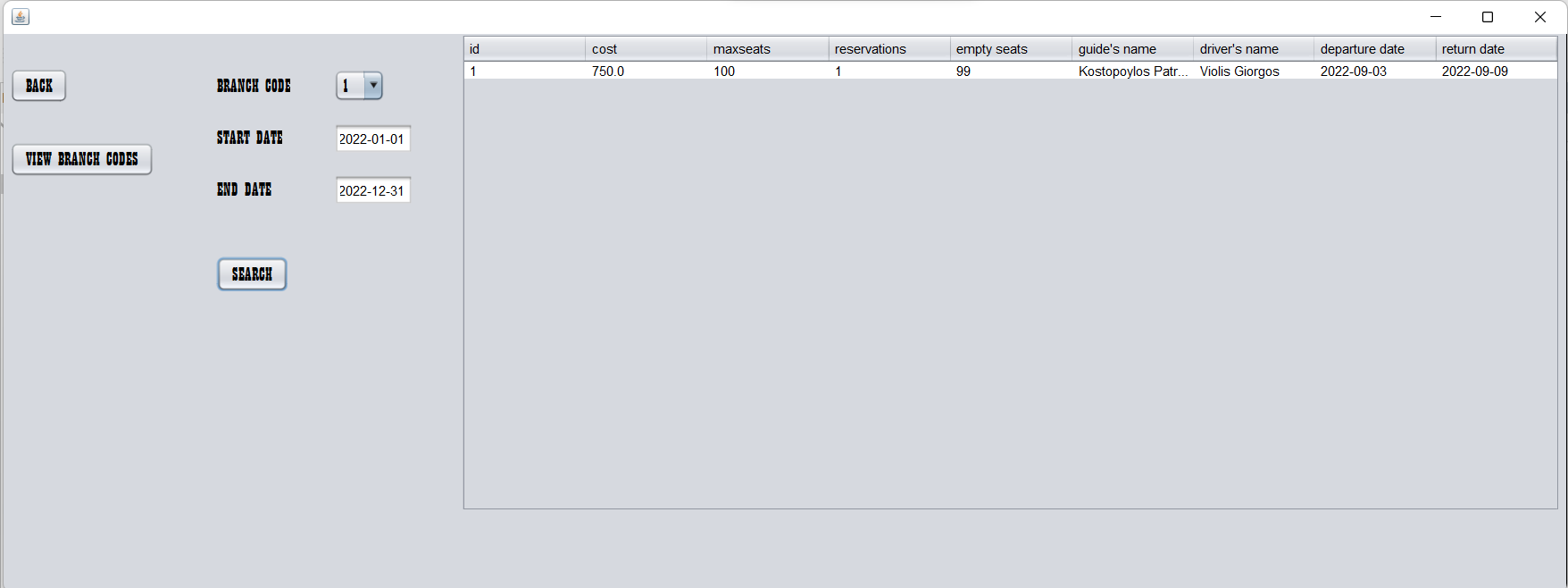
String c10 = String.valueOf(myRs.getString("departure\_date"));

String c11 = String.valueOf(myRs.getString("return\_date"));

String tbData[] = { c1, c2, c3, c4, c5, c6, c7, c10, c11 };

tb1Model.addRow(tbData); }

Στο παράδειγμά μας θα κάνουμε αναζήτηση στο υποκατάστημα 1, για όλα τα ταξίδια που έγιναν μέσα στο 2022.



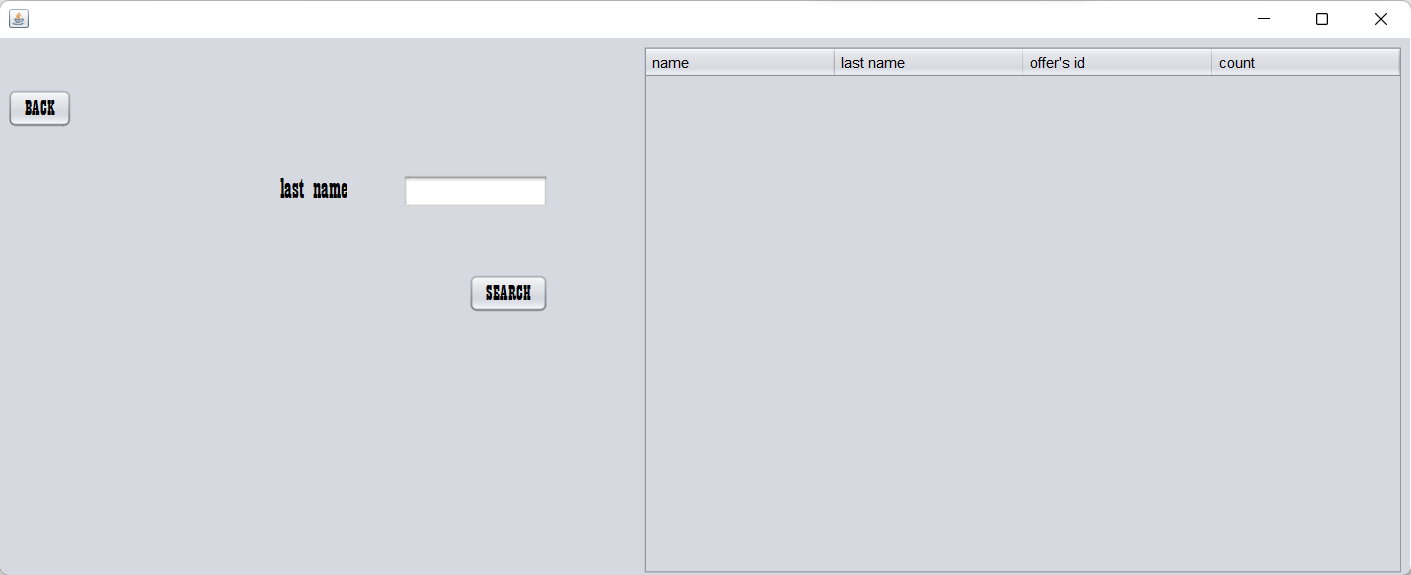
Όπως βλέπουμε, πραγματοποιήθηκε μόνο ένα ταξίδι, του οποίου οι πληροφορίες εμφανίζονται στον πίνακα.

## 4.11 Ζητούμενο για αναζήτηση κρατήσεων

Στο ζητούμενο αυτό, δίνοτας ένα επώνυμο θέλουμε να επιστραφούν όλες οι κρατήσεις που έχουν γίνει στον πίνακα reservation\_offers με το επώνυμο αυτό. Τα αποτελέσματα θα ομαδοποιηθούν ανά προσφορά ταξιδιού (offer, rsof\_of\_id) και επίσης θα καταμετρηθούν οι κρατήσεις με το ίδιο όνομα σε κάθε προσφορά ταξιδιού.

Για να εκτελέσουμε την αναζήτηση επιλέγουμε από το μενού με την σειρά τις καρτέλες

search -> rservation\_offers -> by surname και θα εμφανιστεί το εξής παράθυρο:



Για να εκτελέσουμε την αναζήτηση θα γράψουμε το επώνυμο που θέλουμε στο πεδίο last name και θα πατήσουμε search.

Με το που πατήσουμε search, θα εκτελεστούν οι εξής εντολές:

st = myConn.createStatement();

String sql = "SELECT resof\_name, resof\_lname, COUNT(\*), resof\_of\_id FROM reservation\_offers "

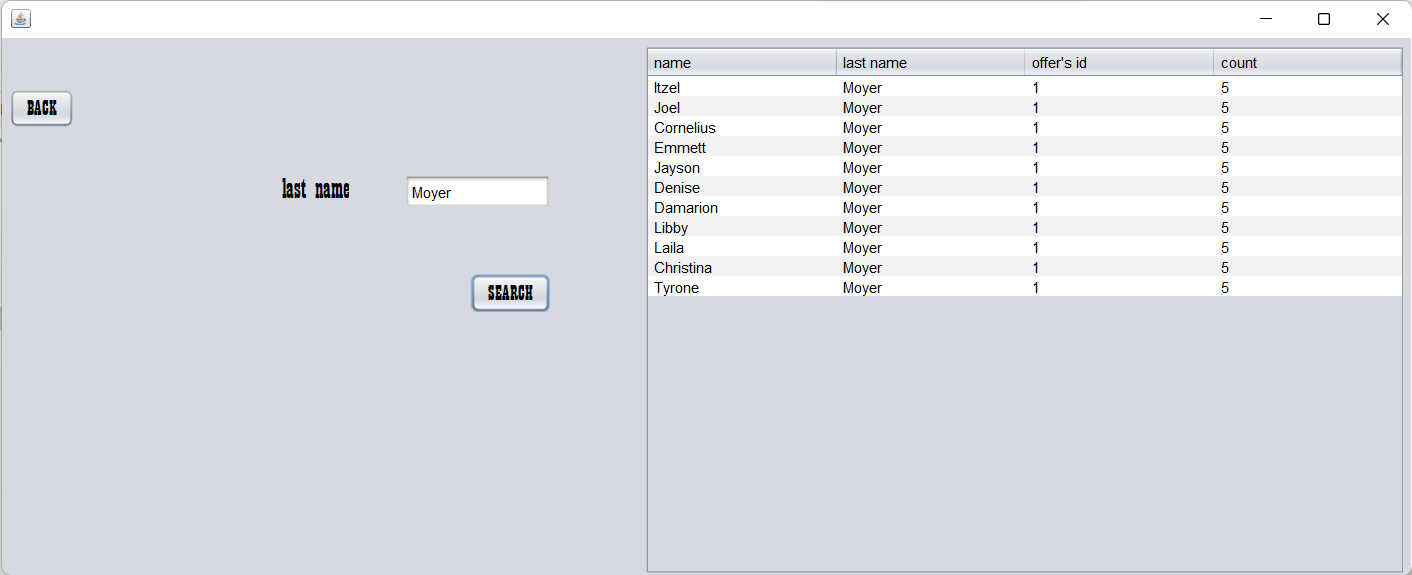
+ "WHERE TRIM(TRAILING '\\r' FROM resof\_lname) = '" + lname + "' "

+ "GROUP BY resof\_of\_id, resof\_name, resof\_lname;";

myRs = st.executeQuery(sql);

Δηλαδή, θα δημιουργηθεί σύνδεση με την βάση δεδομένων μας και θα εκτελεστεί το παραπάνω query.

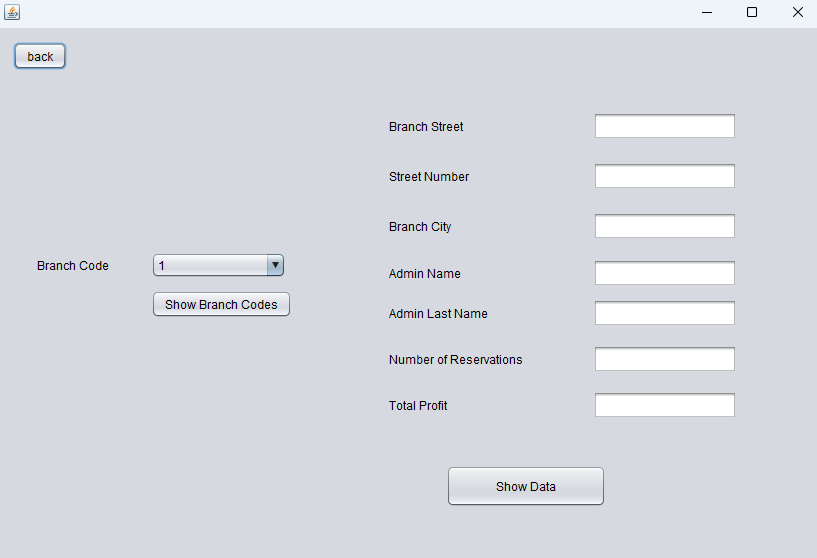
Στο παράδειγμά μας θα ψάξουμε για κρατήσεις στο επίθετο “Moyer”



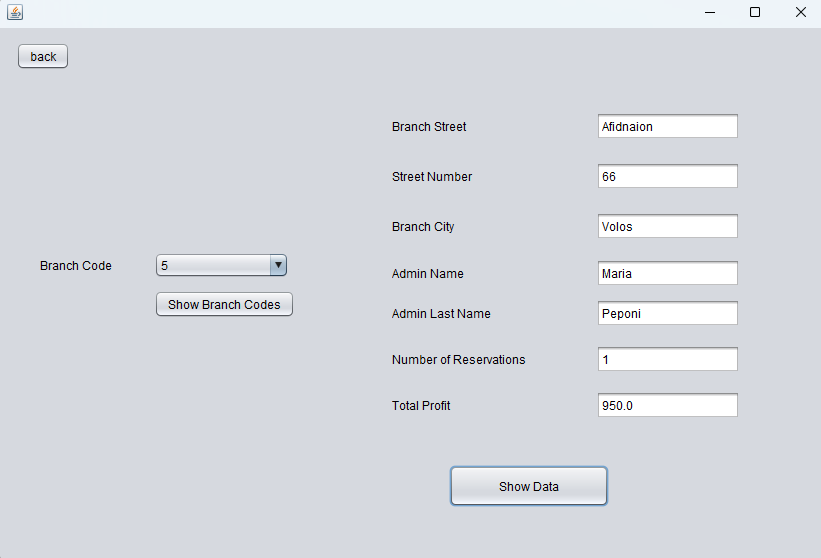
Όπως βλέπουμε εμφανίστηκαν όλες οι κρατήσεις που υπάρχουν με το επώνυμο Moyer, καθώς και πληροφορίες για αυτές, όπως το πρώτο όνομα, το επίθετο, το id της προσφοράς στην οποία έγινε η κράτηση καθώς και πόσες κρατήσεις έγιναν με το ίδιο όνομα.

## 4.12 Ζητούμενο για ένδειξη συνόλου κρατήσεων και εσόδων

Για το συγκεκριμένο ζητούμενο, για να ανοίξει το ανάλογο παράθυρο, πρέπει στο μενού να πατήσουμε το κουμπί “Branch/Profits” δίπλα από την περιγραφή “Branch and Profits Details” το οποίο μας ανοίγει το εξής παράθυρο:



Το παράθυρο αυτό, περιέχει αρχικά όπως και τα υπόλοιπα παράθυρα ένα κουμπί back μέσω του οποίου επιστρέφουμε στο μενού, ένα κουμπί με ονομασία “Show Branch Codes” το οποίο μας εμφανίζει στο combo box όλους τους κωδικούς υποκαταστημάτων που έχουμε και τέλος το κουμπί “Show Data” που εμφανίζει τα ζητούμενα δεδομένα, καθώς και το σύνολο των εσόδων από τις κρατήσεις. Για παράδειγμα, στο υποκατάστημα με κωδικό 5 έχουμε τα εξής δεδομένα:



Για να σιγουρευτούμε ότι τα δεδομένα μας είναι σωστά, μπορούμε να συγκρίνουμε τα αποτελέσματα με την βάση μας όπου εκτελώντας τον εξής κωδικό:

SELECT br\_code, br\_street, br\_num, br\_city, wrk\_name, wrk\_lname, COUNT(res\_tr\_id), SUM(tr\_cost)

FROM worker

JOIN branch ON wrk\_br\_code = br\_code

JOIN admin ON adm\_AT = wrk\_AT

JOIN trip ON tr\_br\_code = br\_code

JOIN reservation ON res\_tr\_id = tr\_id

WHERE br\_code = 5;

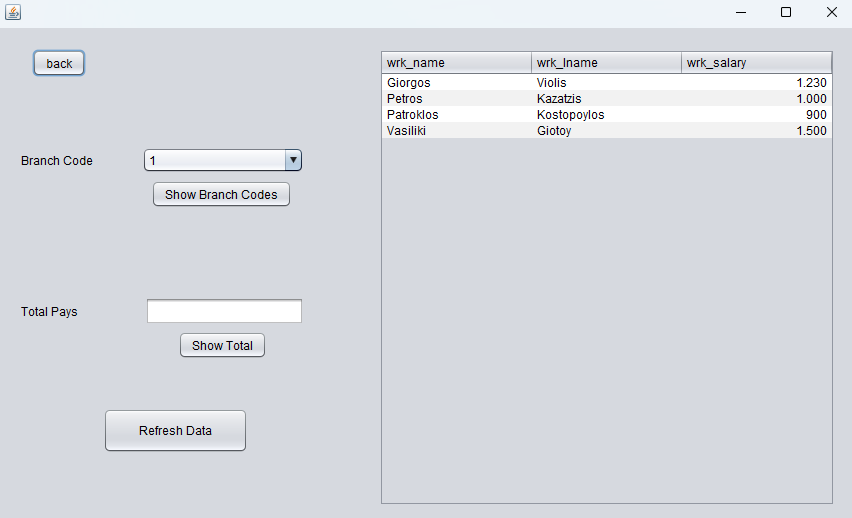
Θα εμφανιστεί στην οθόνη μας το εξής:



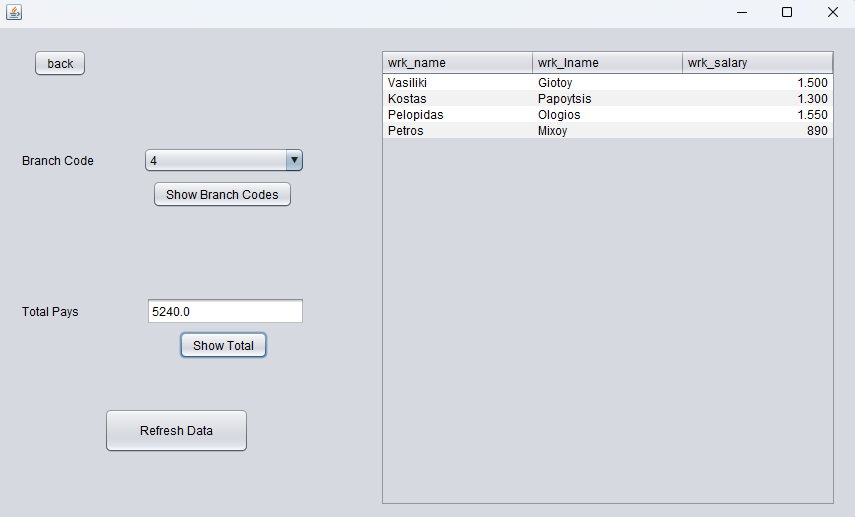
Οπότε τα αποτελέσματά μας είναι σωστά.

## 4.13 Ζητούμενο για ένδειξη πληροφοριών σχετικά με το υποκατάστημα και τους μισθούς

Για το συγκεκριμένο ζητούμενο, για να ανοίξει το ανάλογο παράθυρο, πρέπει στο μενού να πατήσουμε το κουμπί “Branch/Pays” δίπλα από την περιγραφή “Branch and Pays Details” το οποίο μας ανοίγει το εξής παράθυρο:



Όπως παρατηρούμε, αρχικά στα δεξιά του παραθύρου, εμφανίζεται ένας πίνακας ο οποίος περιέχει τα ονόματα, τα επώνυμα και τους μισθούς των υπαλλήλων του υποκαταστήματος που επιλέγει ο χρήστης. Υπάρχει το κουμπί “Show Branch Codes” μέσω του οποίου εμφανίζονται όλοι οι διαθέσιμοι κωδικοί υποκαταστημάτων και το κουμπί “Refresh Data” το οποίο πρέπει να πατηθεί κάθε φορά που επιλέγεται νέο υποκατάστημα για να εμφανίσει τα νέα δεδομένα στον πίνακα. Τέλος, έχουμε φυσικά το κουμπί back μέσω του οποίου επιστρέφουμε στο menu και το κουμπί “Show Total”, μέσω του οποίου εμφανίζεται το άθροισμα των μισθών όλων των υπαλλήλων μας. Αν για παράδειγμα επιλέξουμε το υποκατάστημα με κωδικό 4 για να δούε τις παραπάνω πληροφορίες, θα έχουμε τα εξής αποτελέσματα:



Πράγματι, για να βεβαιωθούμε ότι τα αποτελέσματά μας είναι σωστά, μπορούμε να διασταυρώσουμε τα αποτελέσματά μας εκτελώντας στην MySQL τον παρακάτω κώδικα:

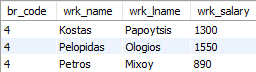
SELECT br\_code, wrk\_name, wrk\_lname, wrk\_salary

FROM worker

JOIN branch ON br\_code = wrk\_br\_code

WHERE br\_code = 4;

Όπου παίρνουμε τα εξής αποτελέσματα:



Παρατηρούμε ότι η “Vasiliki Giotoy” δεν εμφανίζεται στα αποτελέσματα, γιατί είναι η υπεύθυνος IT, που σημαίνει ότι αποτελεί υπάλληλο όλων των υποκαταστημάτων. Από τις εγγραφές, παρατηρούμε ότι ο μισθός της είναι 1500 ευρώ.



Οπότε, 1300+1550+890+1500 = 5240. Οπότε, τα αποτελέσματά μας είναι σωστά.

## 4.17 EXTRA ζητούμενα

Στα ζητούμενα αποφασίσαμε να προσθέσουμε τις εξής λειτουργικότητες:

Φτιάξαμε μια νέα καρτέλα ACCOUNTINGS, στην οποία επιλέγοντας τον κωδικό του υποκαταστήματος που θέλουμε, εμφανίζονται τα έσοδα, τα έξοδα, τα καθαρά έσοδα, ο μέσος μισθός υπαλλήλου, ο μέγιστος μισθός και ο ελάχιστος.

Οι κώδικες των εσόδων και εξόδων έχουν υλοποιηθεί και επεξηγηθεί στα υποερωτήματα 4.3 και 4.4.

Για τα καθαρά έσοδα, αφαιρέσαμε τα έξοδα από τα έσοδα.

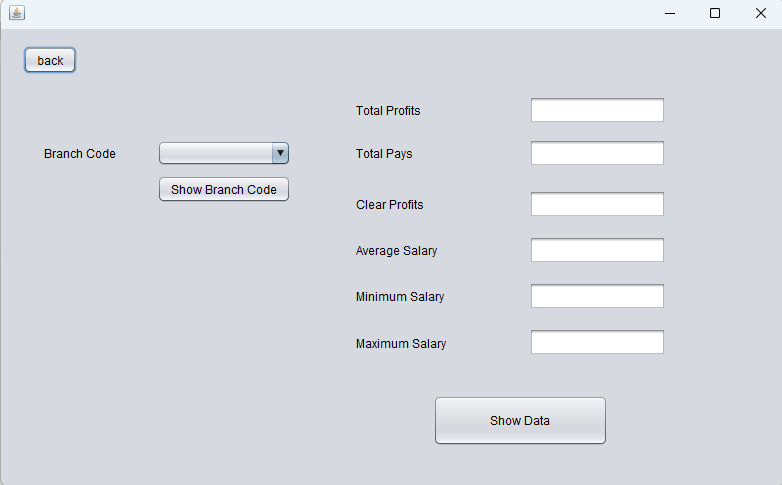
Για τα υπόλοιπα τρια πεδία, εκτελέσαμε το query:

"SELECT AVG(wrk\_salary), MAX(wrk\_salary), MIN(wrk\_salary) FROM worker WHERE wrk\_br\_code="+brc;

Όπου brc ο επιλεγμένος κωδικός του υποκαταστήματος. Έτσι με τις συναρτήσεις της sql AVG(), MAX() και MIN() βρήκαμε τον μέσο όρο, την μέγιστη και την ελάχιστη τιμή της στήλης wrk\_salary από τον πίνακα worker.

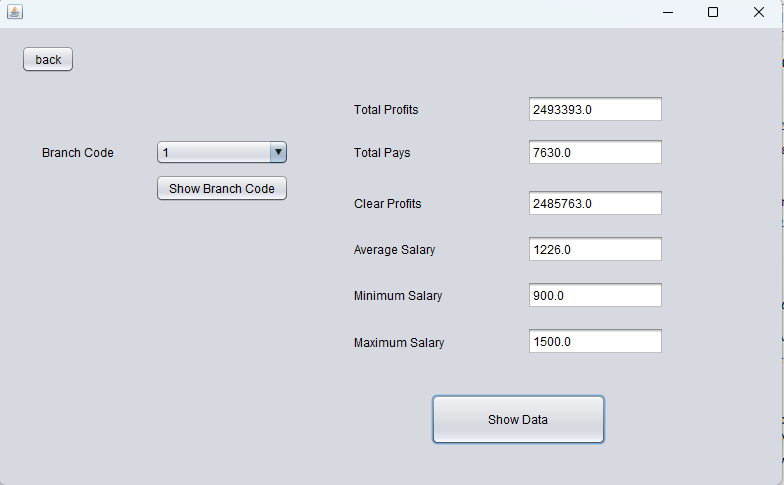
Ας δούμε τώρα ένα παράδειγμα χρήσης της καρτέλας ACCOUNTINGS.

Επιλέγουμε από το μενού την καρτέλα ACCOUNTINGS και εμφανίζεται το εξής παράθυρο:



Πατάμε το κουμπί SHOW BRANCH CODES για να φορτώσουν οι κωδικοί των υποκαταστημάτων και στην συνέχεια επιλέγουμε το υποκατάστημα 1.

Για να εμφανιστούν οι πληροφορίες που θέλουμε, πατάμε το κουμπί SHOW DATA, και αμέσως βλέπουμε τα δεδομένα στα πεδία.



Οι πληροφορίες εμφανίζονται σωστά.