**3.1.1 Προπαρασκευαστικό κομμάτι:**

DROP DATABASE if exists travel\_agency;

CREATE DATABASE travel\_agency;

USE travel\_agency;

CREATE TABLE branch(

br\_code INT(11),

br\_street VARCHAR(30) DEFAULT 'unknown' NOT NULL,

br\_num INT(4) NOT NULL,

br\_city VARCHAR(30) DEFAULT 'unknown' NOT NULL,

PRIMARY KEY (br\_code)

);

CREATE TABLE phones(

ph\_br\_code INT(11) NOT NULL,

ph\_number CHAR(10) NOT NULL,

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

CONSTRAINT BRPHONE

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE worker(

wrk\_AT CHAR(10) NOT NULL,

wrk\_name VARCHAR(20) DEFAULT 'unknown' NOT NULL,

wrk\_lname VARCHAR(20) DEFAULT 'unknown' NOT NULL,

wrk\_salary FLOAT(7,2) NOT NULL,

wrk\_br\_code INT(11) NOT NULL,

PRIMARY KEY (wrk\_AT),

CONSTRAINT BRWORKER

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE admin(

adm\_AT CHAR(10) NOT NULL,

adm\_type ENUM('LOGISTICS','ADMINISTRATIVE','ACCOUNTING') NOT NULL,

adm\_diploma VARCHAR(200) NOT NULL,

PRIMARY KEY (adm\_AT),

CONSTRAINT WORKADMIN

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE manages(

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

mng\_br\_code INT(11) NOT NULL,

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

CONSTRAINT ADMMANAGE

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

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT BRMANAGE

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE driver(

drv\_AT CHAR(10) NOT NULL,

drv\_license ENUM('A','B','C','D') NOT NULL,

drv\_route ENUM('LOCAL','ABROAD') NOT NULL,

drv\_experience TINYINT(4) NOT NULL,

PRIMARY KEY (drv\_AT),

CONSTRAINT WORKDRIVER

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE guide(

gui\_AT CHAR(10) NOT NULL,

gui\_cv TEXT NOT NULL,

PRIMARY KEY (gui\_AT),

CONSTRAINT WORKGUIDE

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

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 GUILANGUAGE

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

);

CREATE TABLE trip(

tr\_id INT(11) NOT NULL,

tr\_departure DATETIME NOT NULL,

tr\_return DATETIME NOT NULL,

tr\_maxseats TINYINT(4) NOT NULL,

tr\_cost FLOAT(7,2) NOT NULL,

tr\_br\_code INT(11) NOT NULL,

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

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

PRIMARY KEY(tr\_id),

CONSTRAINT BRANCHTRIP

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

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT GUIDETRIP

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

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT DRIVERTRIP

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE event(

ev\_tr\_id INT(11) NOT NULL,

ev\_start DATETIME NOT NULL,

ev\_end DATETIME NOT NULL,

ev\_descr TEXT NOT NULL,

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

CONSTRAINT EVENTTRIP

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE reservation(

res\_tr\_id INT(11) NOT NULL,

res\_seatnum TINYINT(4) NOT NULL,

res\_name VARCHAR(20) DEFAULT 'unknown' NOT NULL,

res\_lname VARCHAR(20) DEFAULT 'unknown' NOT NULL,

res\_isadult ENUM('ADULT','MINOR') NOT NULL,

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

CONSTRAINT TRIPRESERV

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

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE destination (

dst\_id INT(11) NOT NULL,

dst\_name VARCHAR(50) NOT NULL,

dst\_descr TEXT NOT NULL,

dst\_rtype ENUM('LOCAL','ABROAD') NOT NULL,

dst\_language VARCHAR(30) NOT NULL,

dst\_location INT(11) ,

PRIMARY KEY(dst\_id),

CONSTRAINT DESTLOCATION

FOREIGN KEY (dst\_location) REFERENCES destination(dst\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE travel\_to(

to\_tr\_id INT(11) NOT NULL,

to\_dst\_id INT(11) NOT NULL,

to\_arrival DATETIME NOT NULL,

to\_departure DATETIME NOT NULL,

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

CONSTRAINT TRIPTRAVEL

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

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT TRAVELDEST

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

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO branch VALUES

(1,'AGIOY ANDREOY','15','PATRA'),

(2,'EVELPIDWN','29','ATHINA'),

(3,'TSIMISKI','43','THESALONIKI');

INSERT INTO phones VALUES

(1,'2610659237'),

(1,'6946318647'),

(2,'2103262024'),

(2,'6932169687'),

(3,'2316052340'),

(3,'6923168945');

INSERT INTO worker VALUES

('DR423611','GIANNIS','PAPADOPOYLOS',850.00,1),

('DR383152','VASILIS','APOSTOLOY',872.25,1),

('DR521363','ANTONIA','OIKONOMOY',800.32,1),

('DR210364','MARIA','PAPANTONIOY',840.00,2),

('DR235065','ANTONIS','ALEXANDROPOYLOS',864.59,2),

('DR464236','GIANNIS','SIFAKIS',967.00,2),

('DR325067','ZOI','PAPADOPOYLOY',1053.34,3),

('DR325068','KONSTANTINA','PAPADOPOYLOY',1099.00,3),

('DR325069','LIANA','PAPAGEORGIOU',980.00,3),

('AD356581','KONSTANTINA','PATAKI',1000.00,1),

('AD736892','ALEXIS','OIKONOMOY',856.41,1),

('AD875913','ANDREAS','PAPPAS',790.36,1),

('AD512354','MARIOS','NIKOLOPOYLOS',868.53,2),

('AD321025','ANASTASIA','ALEXOPOYLOY',980.79,2),

('AD456736','GIORGOS','GEORGAKOPOYLOS',800.00,2),

('AD765667','MARINA','NINIS',832.76,3),

('AD765668','MARIA','GEORGOYLI',835.00,3),

('AD609519','ANDREAS','SOTIROPOYLOS',963.00,3),

('GU648981','DIMITRIS','PAPASTATHOPOYLOS',850.65,1),

('GU435342','VASILIS','APOSTOLOPOYLOS',852.87,1),

('GU678673','APOSTOLHS','MPOYGAS',750.87,1),

('GU768564','MANOS','KYRIAKOPOYLOS',900.50,2),

('GU128745','ALEXIS','GEORGOPOYLOS',890.70,2),

('GU986816','TASOS','GIANOYKAS',900.60,2),

('GU761207','GIORGOS','GIAMEOS',900.60,3),

('GU136088','ATHANASIA','ANTONIOY',810.00,3),

('GU394459','GEWRGIA','ATHANASOPOYLOY',785.63,3);

INSERT INTO admin VALUES

('AD356581','ACCOUNTING','DIPLOMA OIKONOMIKON'),

('AD736892','ADMINISTRATIVE','DIPLOMA DIOIKISIS EPIXEIRISEON'),

('AD875913','LOGISTICS','DIPLOMA LOGISTIKIS'),

('AD512354','ADMINISTRATIVE','DIPLOMA DIOIKISIS EPIXEIRISEON'),

('AD321025','ACCOUNTING','DIPLOMA OIKONOMIKON'),

('AD456736','LOGISTICS','DIPLOMA LOGISTIKIS'),

('AD765667','LOGISTICS','DIPLOMA LOGISTIKIS');

INSERT INTO manages VALUES

('AD356581',1),

('AD512354',2),

('AD765667',3);

INSERT INTO guide VALUES

('GU648981','cv 1'),

('GU435342','cv 2'),

('GU678673','cv 3'),

('GU768564','cv 4'),

('GU128745','cv 5'),

('GU986816','cv 6'),

('GU761207','cv 7'),

('GU136088','cv 8'),

('GU394459','cv 9');

INSERT INTO languages VALUES

('GU648981','English,Spanish'),

('GU435342','English,French'),

('GU678673','English,German'),

('GU768564','English,Spanish'),

('GU128745','English,Spanish'),

('GU986816','English,German'),

('GU761207','English,German'),

('GU136088','English,Spanish'),

('GU394459','English,French');

INSERT INTO driver VALUES

('DR423611','D','LOCAL',12),

('DR383152','A','ABROAD',24),

('DR521363','C','LOCAL',29),

('DR210364','D','ABROAD',31),

('DR235065','A','ABROAD',50),

('DR464236','B','LOCAL',48),

('DR325067','D','LOCAL',60),

('DR325068','A','ABROAD',27),

('DR325069','D','LOCAL',9);

INSERT INTO trip VALUES

(1,'2023-01-05 12:00:00','2023-01-11 14:00:00',40,60.00,1,'GU648981','DR423611'),

(2,'2022-12-22 08:30:00','2022-12-26 21:30:00',36,30.00,1,'GU435342','DR383152'),

(3,'2023-02-09 09:00:00','2023-02-17 10:00:00',40,42.63,2,'GU768564','DR210364'),

(4,'2023-02-25 10:45:00','2023-03-01 12:15:00',42,28.79,3,'GU761207','DR325067'),

(5,'2023-01-17 13:15:00','2023-01-22 15:00:00',39,40.00,3,'GU136088','DR325068');

INSERT INTO event VALUES

(1,'2023-01-06 12:00:00','2023-01-07 17:00:00','Event 1.1 description'),

(1,'2023-01-08 08:30:00','2023-01-08 22:30:00','Event 1.2 description'),

(1,'2023-01-09 10:00:00','2023-01-10 14:00:00','Event 1.3 description'),

(2,'2022-12-23 10:40:00','2022-12-24 19:15:00','Event 2.1 description'),

(2,'2022-12-25 16:00:00','2022-12-26 14:45:00','Event 2.2 description'),

(3,'2023-02-10 10:30:00','2023-02-11 02:00:00','Event 3.1 description'),

(3,'2023-02-12 08:00:00','2023-02-13 16:00:00','Event 3.2 description'),

(3,'2023-02-14 12:15:00','2023-02-15 12:00:00','Event 3.3 description'),

(3,'2023-02-15 15:30:00','2023-02-16 20:00:00','Event 3.4 description'),

(4,'2023-02-26 16:15:00','2023-02-27 19:45:00','Event 4.1 description'),

(4,'2023-02-28 08:00:00','2023-02-28 19:45:00','Event 4.2 description'),

(5,'2023-01-18 09:00:00','2023-01-19 13:00:00','Event 5.1 description'),

(5,'2023-01-20 10:00:00','2023-01-21 17:00:00','Event 5.2 description');

INSERT INTO reservation VALUES

(1,1,'Paulos','Giannopoulos','ADULT'),

(1,2,'Thodoris','Stefanopoylos','ADULT'),

(1,3,'Ioanna','Stefanopoylou','MINOR'),

(1,4,'Stefanos','Giannakis','ADULT'),

(1,5,'Petros','Giannakis','MINOR'),

(2,1,'Aggeliki','Papandreoy','ADULT'),

(2,2,'Fotini','Papandreoy','MINOR'),

(2,3,'Athanasia','Apostolopoyloy','ADULT'),

(2,4,'Stela','Apostolopoyloy','MINOR'),

(2,5,'Thanasis','Aggelopoylos','ADULT'),

(2,6,'Thodora','Mixailidi','ADULT'),

(3,1,'Apostolis','Aggelopoylos','MINOR'),

(3,2,'Alexandra','Melanidoy','ADULT'),

(3,3,'Apostolia','Melanidoy','MINOR'),

(3,4,'Tasos','Sakos','ADULT'),

(3,5,'Eleonora','Sakoy','MINOR'),

(3,6,'Dimitrhs','Papanikolaos','ADULT'),

(3,7,'Pantelis','Georgakopoylos','ADULT'),

(3,8,'Anastasia','Papanikolaou','MINOR'),

(3,9,'Menelaos','Papaloykas','ADULT'),

(4,1,'Pantelis','Giannopoulos','ADULT'),

(4,2,'Kwstas','Stefanopoylos','ADULT'),

(4,3,'Ioanna','Mixalopoyloy','ADULT'),

(4,4,'Orestis','Giannakis','ADULT'),

(4,5,'Thanasis','Giannopoulos','MINOR'),

(5,1,'Pinelopi','Papandreoy','ADULT'),

(5,2,'Fotini','Mixalopoyloy','ADULT'),

(5,3,'Athanasia','Apostoloy','ADULT'),

(5,4,'Stefania','Aggelopoyloy','MINOR'),

(5,5,'Marios','Aggelopoylos','ADULT'),

(5,6,'Domna','Mixailidi','ADULT'),

(5,7,'Apostolis','Metaxas','MINOR'),

(5,8,'Alexia','Metaxa','ADULT'),

(5,9,'Apostolia','Papadopoyloy','MINOR'),

(5,10,'Tasos','Papadopoylos','ADULT'),

(5,11,'Elenh','Papadopouloy','MINOR'),

(5,12,'Dimitra','Papanikolaoy','ADULT'),

(5,13,'Andreas','Georgakopoylos','ADULT'),

(5,14,'Anastasia','Papaloyka','MINOR'),

(5,15,'Anastasis','Papaloykas','ADULT');

INSERT INTO destination VALUES

(1,'USA','Destination 1 description','ABROAD','English',NULL),

(2,'Greece','Destination 2 description','LOCAL','Greek',NULL),

(3,'New York','Destination 3 description','ABROAD','English',1),

(4,'Los Angeles','Destination 4 description','ABROAD','English',1),

(5,'Crete','Destination 5 description','LOCAL','Greek',2);

INSERT INTO travel\_to VALUES

(1,1,'2023-01-05 22:00:00','2023-01-11 04:00:00'),

(2,2,'2022-12-22 11:30:00','2022-12-26 18:30:00'),

(3,3,'2023-02-09 18:00:00','2023-02-17 01:00:00'),

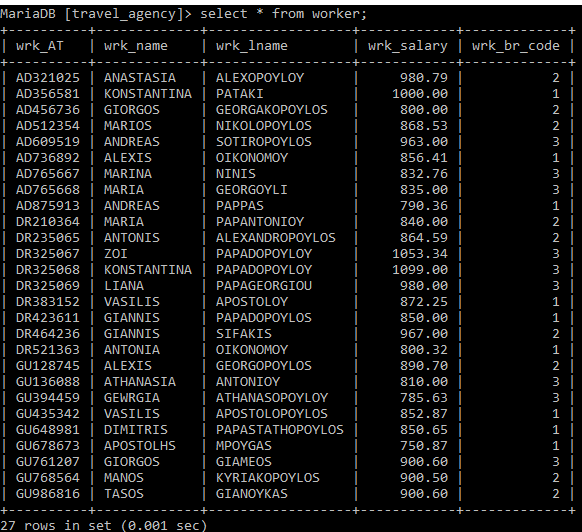
(4,4,'2023-02-25 22:45:00','2023-03-01 00:15:00'),

(5,5,'2023-01-17 16:15:00','2023-01-22 12:00:00');

**3.1.2 ΝΕΕΣ απαιτήσεις**

**3.1.2.1**

**Αρχικός πίνακας worker:**



**Εισαγωγή υπεύθυνων πληροφορικής στον πίνακα worker:**

INSERT INTO worker Values

('IT326491','DIONYSIS','KALAITZAKIS', 1000.00, 1),

('IT315482','KYRIAKI','KEFALA',1125.00,2),

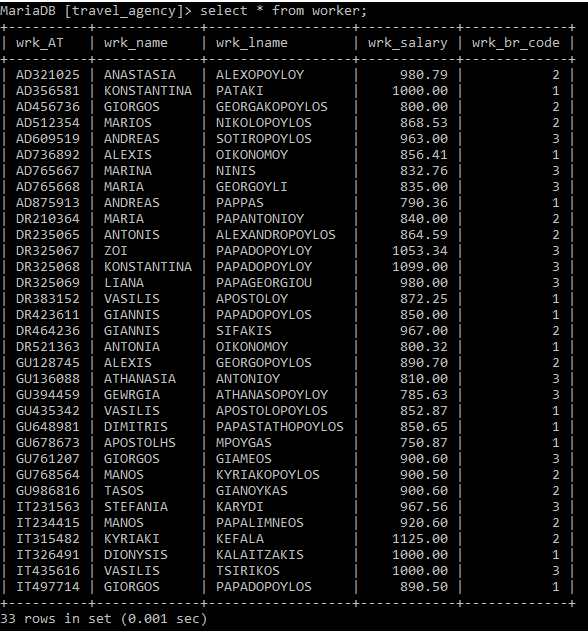
('IT231563','STEFANIA','KARYDI',967.56,3),

('IT497714','GIORGOS','PAPADOPOYLOS',890.50,1),

('IT234415','MANOS','PAPALIMNEOS',920.60,2),

('IT435616','VASILIS','TSIRIKOS',1000.00,3);

**Πίνακας worker μετά από εισαγωγές:**



**Δημιουργία πίνακα υπεύθυνων πληροφορικής:**

CREATE TABLE it(

IT\_AT CHAR(10) NOT NULL,

IT\_password CHAR(10) DEFAULT 'password' NOT NULL,

IT\_start\_date DATETIME NOT NULL,

IT\_end\_date DATETIME,

PRIMARY KEY (IT\_AT),

CONSTRAINT ITWORKER

FOREIGN KEY (IT\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

**Εισαγωγή δεδομένων στον πίνακα it:**

INSERT INTO it Values

('IT326491',DEFAULT,'2013-05-18 00:00',NULL),

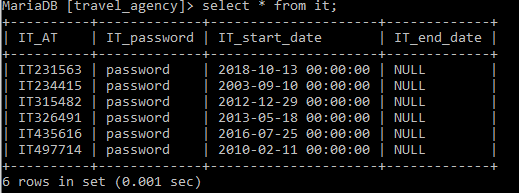
('IT315482',DEFAULT,'2012-12-29 00:00',NULL),

('IT231563',DEFAULT,'2018-10-13 00:00',NULL),

('IT497714',DEFAULT,'2010-02-11 00:00',NULL),

('IT234415',DEFAULT,'2003-09-10 00:00',NULL),

('IT435616',DEFAULT,'2016-07-25 00:00',NULL);



**3.1.2.2**

**Δημιουργία πίνακα log:**

CREATE TABLE log(

log\_AT CHAR(10) NOT NULL,

table\_trip ENUM('INSERT','UPDATE','DELETE'),

trip\_last\_update DATETIME,

table\_reservation ENUM('INSERT','UPDATE','DELETE'),

reservation\_last\_update DATETIME,

table\_event ENUM('INSERT','UPDATE','DELETE'),

event\_last\_update DATETIME,

table\_travel\_to ENUM('INSERT','UPDATE','DELETE'),

travel\_to\_last\_update DATETIME,

table\_destination ENUM('INSERT','UPDATE','DELETE'),

destination\_last\_update DATETIME,

PRIMARY KEY (log\_AT),

CONSTRAINT LOGIT

FOREIGN KEY (log\_AT) REFERENCES it(IT\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

**Παραδοχή: Με την χρήση της εντολής ENUM δείχνουμε ποια δεδομένα μπορούν να εισαχθούν στον πίνακα log. Η κάθε στήλη θα παίρνει το κάθε δεδομένο ανάλογα με την χρήση του υπεύθυνου πληροφορικής. Δηλαδή αν κάποιος από τους IΤs κάνει insert σε κάποιον πίνακα το αντίστοιχο γνώρισμα του log θα πάρει την τιμή “INSERT” κτλπ. Τα γνωρίσματα της μορφής “(table)\_last\_update” δείχνουν την τελευταία στιγμή που έγινε η κάθε είσοδος στον ανάλογο πίνακα από τον IT.** **(Η υλοποίηση θα γίνει σε επόμενο ερώτημα).**

**Εισαγωγή αρχικών δεδομένων στον πίνακα log:**

INSERT INTO log(log\_AT) VALUES

(‘IT326491’),

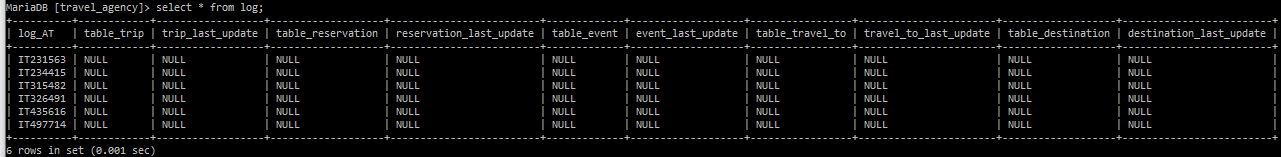
(‘IT315482’),

(‘IT231563’),

(‘IT497714’),

(‘IT234415’),

(‘IT435616’);



**3.1.2.3**

**Δημιουργία πίνακα offers:**

CREATE TABLE offers(

offer\_id INT(11) NOT NULL,

offer\_start\_date DATETIME NOT NULL,

offer\_end\_time DATETIME NOT NULL,

offer\_cost FLOAT(7,2) NOT NULL,

offer\_dst\_id INT(11) NOT NULL,

PRIMARY KEY (offer\_id),

CONSTRAINT OFFERDESTINATION

FOREIGN KEY (offer\_dst\_id) REFERENCES destination(dst\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

**Δημιουργία πίνακα reservation\_offers:**

CREATE TABLE reservation\_offers(

reservation\_id INT(11) NOT NULL,

res\_name VARCHAR(20) NOT NULL,

res\_lastname VARCHAR(20) NOT NULL,

res\_offer\_id INT(11) NOT NULL,

adv\_pay FLOAT(7,2) NOT NULL,

PRIMARY KEY (reservation\_id),

CONSTRAINT RESOFFER

FOREIGN KEY (res\_offer\_id) REFERENCES offers(offer\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

**Εισαγωγή προορισμών προς προσφορά στον πίνακα destination:**

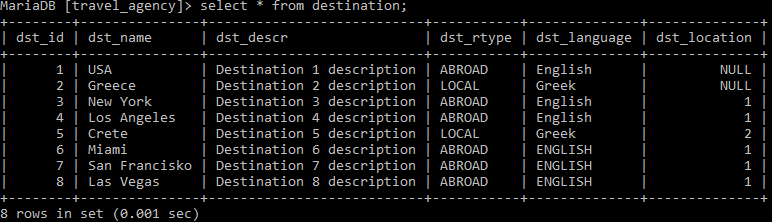
INSERT INTO destination VALUES

(6,'Miami','Destination 6 description','ABROAD','ENGLISH',1),

(7,'San Francisko', 'Destination 7 description','ABROAD','ENGLISH',1),

(8,'Las Vegas','Destination 8 description','ABROAD','ENGLISH',1);

**Πίνακας destination μετά από εισαγωγές:**



**Εισαγωγή προσφορών στον πίνακα offers:**

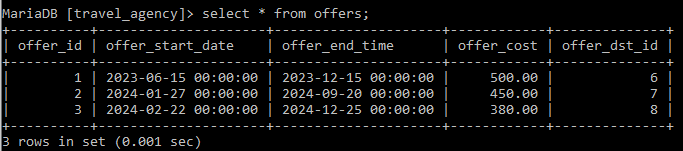
INSERT INTO offers VALUES

(1,’2023-6-15’,’2023-12-15’,500,6),

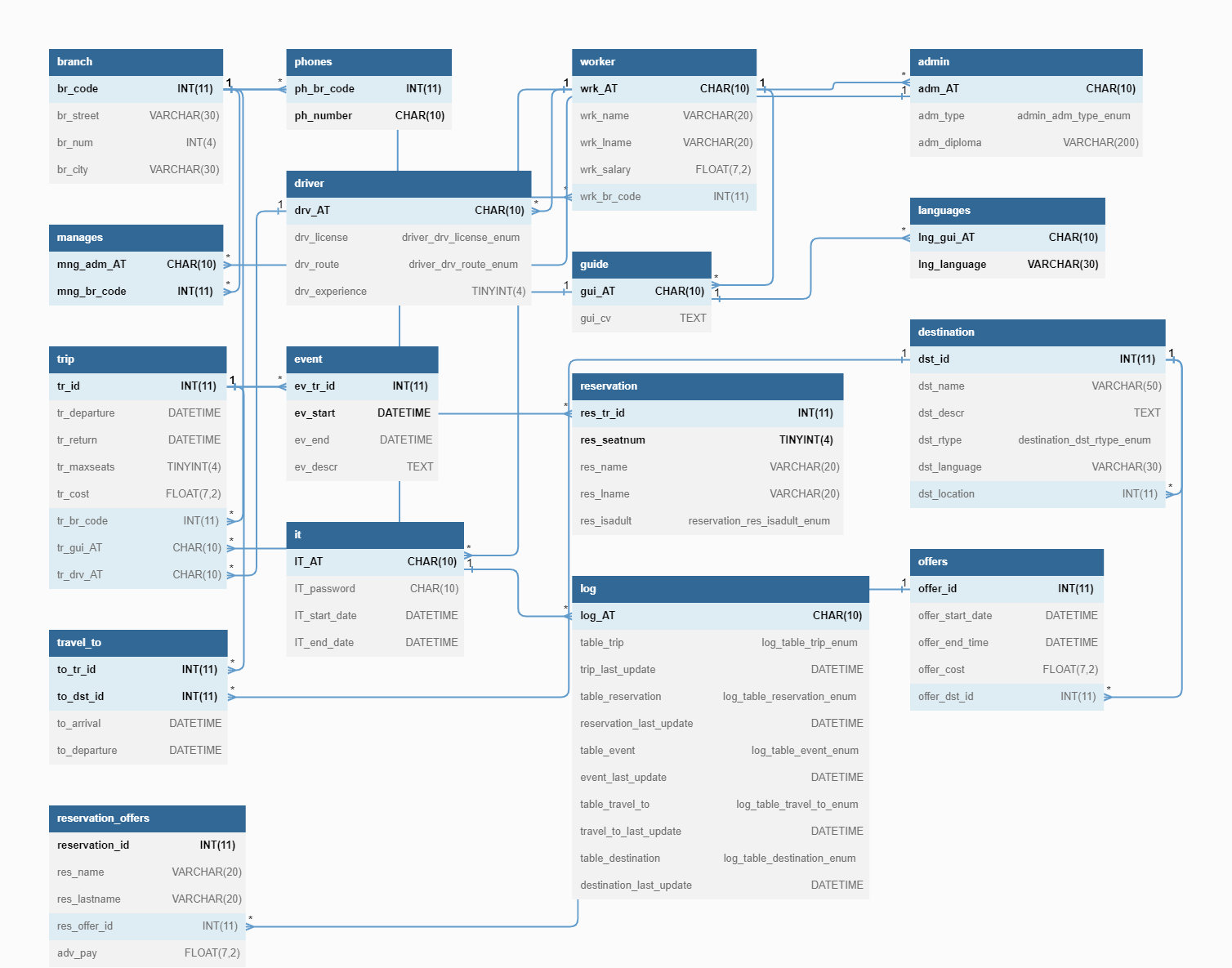
(2,’2024-1-27’,’2024-9-20’,450,7),

(3,’2024-2-22’,’2024-12-25’,380,8);

**Πίνακας offers μετά από εισαγωγές:**



**Τελικό σχεσιακό διάγραμμα:**



**Κώδικας σε python για υλοποίηση των εισαγωγών στον πίνακα reservation\_offers:**

import random

num = 1

while num <= 60000:

namelist = ["Emily", "Madison", "Ashley", "Sarah", "Jessica", "Brianna", "Jordan", "Taylor", "Hannah", "Alexis", "Samantha", "Rachel", "Amber", "Megan", "Alyssa", "Lauren", "Victoria", "Diana", "Maria", "Evelyn", "Elizabeth", "Kayla", "Annabelle", "Mackenzie", "Nicole", "Bailey", "Paige", "Jacqueline", "Caitlin", "Jasmine", "Kaitlyn", "Savannah", "Katherine", "Sofia", "Avery", "Brooklyn", "Eleanor", "Adriana", "Gianna", "Jenna", "Kimberly", "Makayla", "Bella", "Aaliyah", "Allison", "Claire", "Mikayla", "Maria", "Aurora", "Aurora", "Isabella", "Maria", "Aurora", "Sofia", "Isabella", "Mia", "Camila", "Valentina", "Natalia", "Luciana", "Valeria", "Adriana", "Ximena", "Julieta", "Daniela", "Gabriela", "Alondra", "Brenda", "Carmen", "Gabriela", "Liliana", "Paulina", "Miranda", "Dulce", "Esperanza", "Guadalupe", "Karina", "Lourdes", "Rosa", "Salvador", "Adela", "Fatima", "Jazmin", "Jimena", "Juana", "Karen", "Leticia", "Luz", "Magdalena", "Maria", "Maya", "Mireya", "Paula", "Rebeca", "Rocio", "Sandra", "Sofia", "Teresa", "Violeta", "Ai", "Aiko", "Aileen", "Aimee", "Airi", "Akane", "Aki", "Akiko", "Akira", "Ako", "Ami", "Aya", "Ayano", "Eri", "Haruka", "Hikari", "Hina", "Hinata", "Hiroko", "Hitomi", "Honoka", "Hotaru", "Ichiro", "Ikuo", "Jiro", "Kaede", "Kaito", "Kana", "Kanako","Kazuko", "Keiko", "Kimi", "Kosuke", "Makoto", "Mami", "Masako", "Masami", "Maya", "Miyuki", "Nao", "Naoko", "Naomi", "Natsu", "Natsumi", "Noboru", "Noriko", "Reiko", "Rika", "Riku", "Rina", "Sachiko", "Saki", "Sakura", "Satomi", "Takako", "Takashi", "Taro", "Aarav", "Aryan", "Abhinav", "Aditya", "Aryan", "Amit", "Aniket", "Arnav", "Arjun", "Arvind", "Aryan", "Ashwin", "Bhavesh", "Chandan", "Devansh", "Dhruv", "Divyansh", "Gaurav", "Hrithik", "Ishaan", "Ishan", "Kunal", "Kush", "Lakshya", "Naveen", "Nimit", "Nirav", "Parth", "Pranav", "Rishabh", "Rohan", "Sahil", "Saksham", "Sarvesh", "Shivansh", "Shivendra", "Shivansh", "Shreyas", "Siddharth", "Sohan", "Sudhanshu", "Suyash", "Tushar", "Uttam", "Ved", "Vivaan", "Yash", "Yogesh", "Alexander", "Anastasia", "Andrei", "Anna", "Anton", "Dmitri", "Ekaterina", "Elena", "Evgeni", "Galina", "Igor", "Irina", "Kirill", "Konstantin", "Larisa", "Maria", "Mikhail", "Natalia", "Nikolai", "Olga", "Pavel", "Sergei", "Sofia", "Tatiana", "Vadim", "Valeria", "Vasili", "Vera", "Victor", "Vladimir", "Yaroslav", "Yulia", "Yuri", "Zhanna", "Alyona", "Anastasiya", "Andrey", "Anzhelika", "Artyom", "Darya", "Dmitriy", "Egor", "Elizaveta", "Evdokiya", "Grigory", "Inna", "Ivan", "Ksenia", "Mariya", "Maxim", "Mikhail", "Nadezhda", "Natalya", "Nikolay"]

lastnamelist = ["Smith", "Johnson", "Williams", "Jones", "Brown", "Davis", "Miller", "Wilson", "Moore", "Taylor", "Anderson", "Thomas", "Jackson", "White", "Harris", "Martin", "Thompson", "Garcia", "Martinez", "Robinson", "Clark", "Rodriguez", "Lewis", "Lee", "Walker", "Hall", "Allen", "Young", "Hernandez", "King", "Wright", "Lopez", "Hill", "Scott", "Green", "Adams", "Baker", "Gonzalez", "Nelson", "Carter", "Mitchell", "Perez", "Roberts", "Turner", "Phillips", "Campbell", "Parker", "Evans", "Edwards", "Stewart", "Flores", "Morris", "Nguyen", "Garcia", "Gonzalez", "Rodriguez", "Fernandez", "Lopez", "Martinez", "Sanchez", "Perez", "Gomez", "Martín", "Jiménez", "Hernández", "Díaz", "Moreno", "Álvarez", "Muñoz", "Romero", "Alonso", "Gutiérrez", "Navarro", "Torres", "Domínguez", "Vázquez", "Ramos", "Gil", "Ramírez", "Serrano", "Blanco", "Suarez", "Molina", "Morales", "Ortiz", "Rubio", "Marin", "Sanz", "Nuñez", "Iglesias", "Medina", "Garrido", "Cortes", "Castro", "Ortega", "Diez", "Cabrera", "Rey", "Herrera", "Victoria", "Muñoz", "Peña", "Cano", "Lozano", "Castillo", "Kim", "Lee", "Park", "Choi", "Jung", "Kang", "Hong", "Yoon", "Ahn", "Cho", "Lim", "Shim", "Shin", "Yoo", "Yoon", "Wang", "Lin", "Chen", "Chang", "Chou", "Liu", "Huang","Tsai", "Hsu", "Wu", "Lin", "Chu", "Cheng", "Chen", "Liu", "Lo", "Luo", "Zheng", "Zhang", "Zhou", "Xu", "Wu", "Tu", "Tang", "Shi", "Qin", "Pan", "Liang", "Jiang", "Hu", "Feng", "Dong", "Cao", "Chen", "Kumar", "Sharma", "Gupta", "Singh", "Patel", "Kapoor", "Chauhan", "Verma", "Yadav", "Jha", "Ahuja", "Nayar", "Rao", "Khan", "Mehra", "Reddy", "Das", "Malik", "Chopra", "Gandhi", "Bose", "Bhat", "Varma", "Bhushan", "Mahajan", "Prakash", "Seth", "Mathur", "Shankar", "Srivastava", "Agarwal", "Agrawal", "Nair", "Menon", "Shukla", "Tiwari", "Dutt", "Jain", "Dasgupta", "Kaul", "Saraf", "Sarin", "Sethi", "Somaiya", "Suri", "Swamy", "Vaidya", "Vyas", "Mehta", "Tandon", "Nagpal", "Ivanov", "Smirnov", "Kuznetsov", "Popov", "Vasiliev", "Sokolov", "Mikhailov", "Lebedev", "Kovalev", "Grigoriev", "Egorov", "Pavlov", "Morozov", "Zhukov", "Fedorov", "Kozlov", "Novikov", "Konovalov", "Tkachev", "Shchukin", "Shapovalov", "Rozhkov", "Krylov", "Korolev", "Belov", "Volkov", "Solovyev", "Bogdanov", "Andreyev", "Polyakov", "Vinogradov", "Medvedev", "Kozhin", "Lisitsyn", "Koshelev", "Kolosov", "Kolomiytsev", "Khokhlov", "Kazakov", "Kashin", "Kapustin", "Ivanenko", "Gusev", "Danilov", "Chernyshev", "Bryzgalov", "Boyko", "Andrianov", "Aleksandrov"]

reservation\_id = num

res\_name = random.choice(namelist)

res\_lastname = random.choice(lastnamelist)

res\_offer\_id = 3

adv\_pay = random.randint(50, 200)

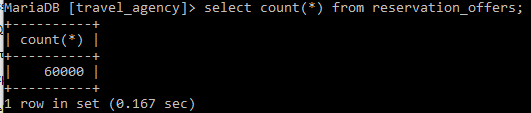
list = (reservation\_id, res\_name, res\_lastname, res\_offer\_id, adv\_pay)

print(list, ',')

num = num + 1

**Εισαγωγή 60000 δεδομένων στον πίνακα reservation\_offers:**

****



**Παρατήρηση 1: Ο κώδικας της insert για τον πίνακα reservation\_offers είναι σε ξεχωριστό αρχείο για εξοικονόμηση χώρου.**

**Παρατήρηση 2: Παρατηρήσαμε ότι κάποια από τα ονόματα που βάλαμε από ένα name generator έχουν τόνους σε λατινικούς χαρακτήρες. Η mysql δεν το αναγνωρίζει αυτό και αντικαθιστά αυτούς τους χαρακτήρες με ?. Εξαιτίας έλλειψης χρόνου όμως δεν προλάβαμε να το διορθώσουμε.**

**3.1.3 Δημιουργία Stored Procedure**

**3.1.3.1**

**Κώδικας procedure για εισαγωγή νέων worker(driver):**

DROP PROCEDURE IF EXISTS new\_driver;

DELIMITER $

CREATE PROCEDURE new\_driver(new\_AT CHAR(10),new\_name VARCHAR(20),new\_lastname VARCHAR(20),new\_salary FLOAT(7,2),new\_license ENUM('A','B','C','D'),new\_route ENUM('LOCAL','ABROAD'),new\_experience TINYINT(4) )

BEGIN

DECLARE num\_of\_drv\_1 INT;

DECLARE num\_of\_drv\_2 INT;

DECLARE num\_of\_drv\_3 INT;

SELECT count(\*) INTO num\_of\_drv\_1

FROM driver INNER JOIN worker on

drv\_AT=wrk\_AT where wrk\_br\_code=1;

SELECT count(\*) INTO num\_of\_drv\_2

FROM driver INNER JOIN worker on

drv\_AT=wrk\_AT where wrk\_br\_code=2;

SELECT count(\*) INTO num\_of\_drv\_3

FROM driver INNER JOIN worker on

drv\_AT=wrk\_AT where wrk\_br\_code=3;

IF(num\_of\_drv\_1<=num\_of\_drv\_2) THEN

IF(num\_of\_drv\_1<=num\_of\_drv\_3) THEN

INSERT INTO worker VALUES(new\_AT, new\_name, new\_lastname, new\_salary, 1);

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

ELSEIF(num\_of\_drv\_3<num\_of\_drv\_1) THEN

INSERT INTO worker VALUES(new\_AT, new\_name, new\_lastname, new\_salary, 3);

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

END IF;

ELSEIF(num\_of\_drv\_2<num\_of\_drv\_1) THEN

IF(num\_of\_drv\_2<=num\_of\_drv\_3) THEN

INSERT INTO worker VALUES(new\_AT, new\_name, new\_lastname, new\_salary, 2);

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

ELSEIF(num\_of\_drv\_3<num\_of\_drv\_2) THEN

INSERT INTO worker VALUES(new\_AT, new\_name, new\_lastname, new\_salary, 3);

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

END IF;

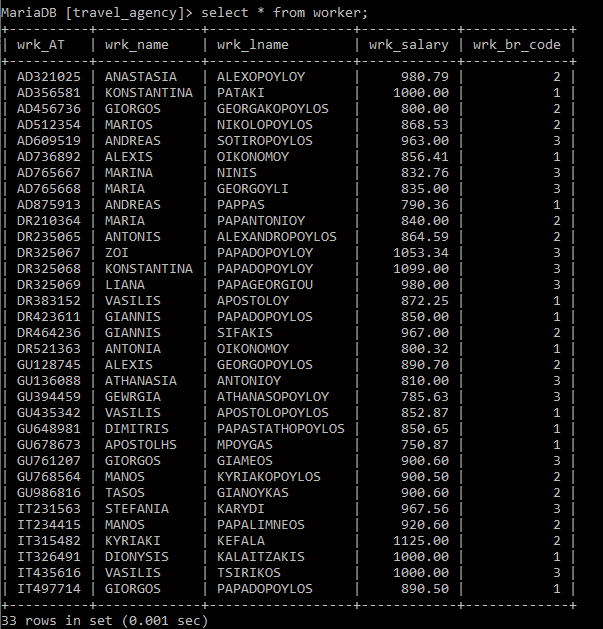
END IF;

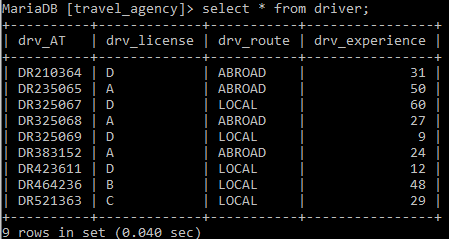
END$

DELIMITER ;

**Παραδοχή: Όταν κάποια παραρτήματα έχουν τον ίδιο αριθμό οδηγών ο νέος οδηγός εγγράφεται στο πρώτο κατά σειρά παράρτημα. Για παράδειγμα όταν όλα τα παραρτήματα έχουν τον ίδιο αριθμό οδηγών ο νέος οδηγός εγγράφεται στο παράρτημα 1 ή όταν το παράρτημα 2 έχει ίδιους οδηγούς με το παράρτημα 3 και λιγότερους από το παράρτημα 1 ο νέος οδηγός εγγράφεται στο παράρτημα 2 κτλπ.**

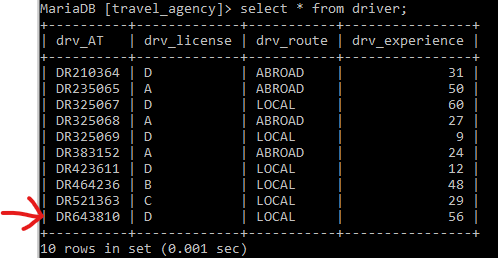
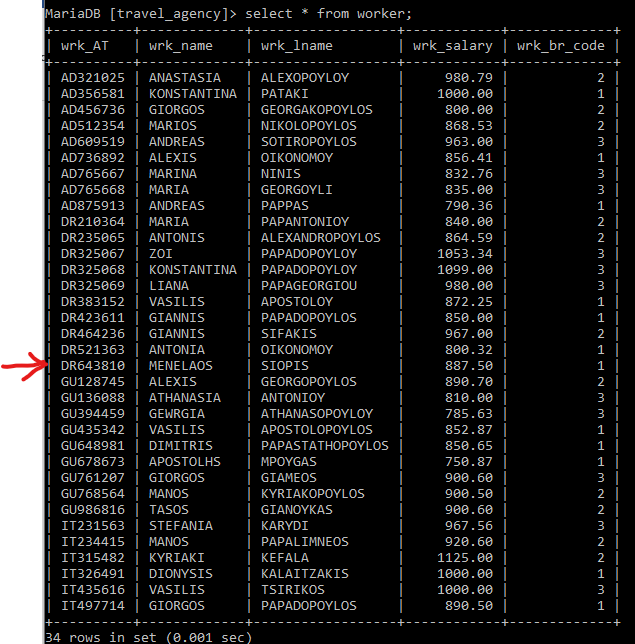
**Αρχικοί πίνακες worker και driver:**



****

**Πίνακες worker και driver μετά από πρώτο παράδειγμα:**

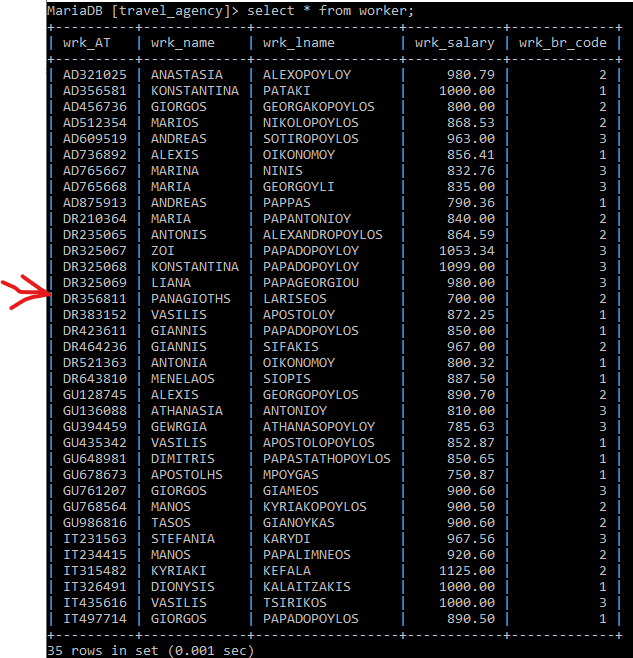
**call new\_driver('DR643810','MENELAOS','SIOPIS',887.5,'D','LOCAL',56);**

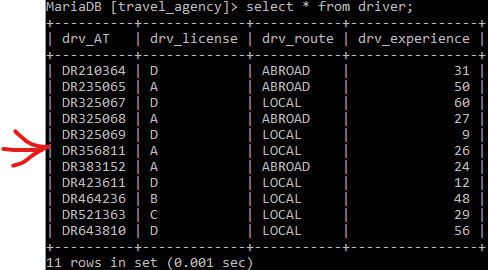


**Όπως φαίνεται αφού όλα τα παραρτήματα έχουν τον ίδιο αριθμό οδηγών, ο νέος οδηγός θα εγγραφεί στο παράρτημα 1.**

**Πίνακες worker και driver μετά από δεύτερο παράδειγμα:**

**Call new\_driver('DR356811','PANAGIOTHS','LARISEOS',700.00,'A','LOCAL',26);**

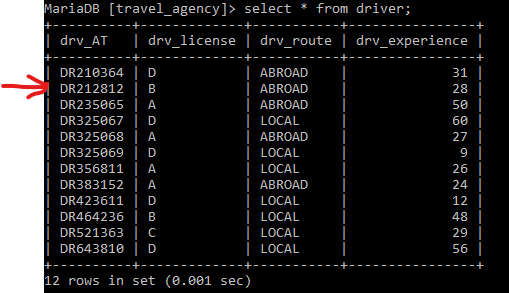
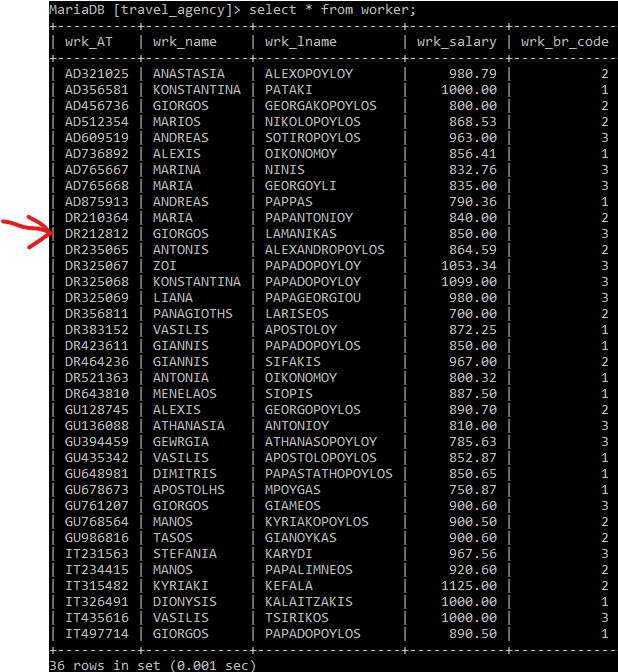


****

**Όπως φαίνεται αφού τα παραρτήματα 2 και 3 έχουν τον ίδιο αριθμό οδηγών και λιγότερο αριθμό οδηγών από το 1, ο νέος οδηγός θα εγγραφεί στο παράρτημα 2.**

**Πίνακες worker και driver μετά από τρίτο παράδειγμα:**

**call new\_driver('DR212812','GIORGOS','LAMANIKAS',850.00,'B','ABROAD',28);**



**Όπως φαίνεται αφού το παράρτημα 3 έχει λιγότερο αριθμό οδηγών από τα 1 και 2, ο νέος οδηγός θα εγγραφεί στο παράρτημα 3.**

**3.1.3.2**

**Κώδικας procedure για εμφάνιση συγκεκριμένου ταξιδιού:**

DROP PROCEDURE IF EXISTS get\_trip;

DELIMITER $

CREATE PROCEDURE get\_trip(br\_trip INT, start\_date DATETIME, end\_date DATETIME)

BEGIN

DECLARE date\_begin DATETIME;

DECLARE res\_seats INT;

DECLARE max\_seats INT;

DECLARE empty\_seats INT;

DECLARE dr\_name VARCHAR(20);

DECLARE dr\_lname VARCHAR(20);

DECLARE gu\_name VARCHAR(20);

DECLARE gu\_lname VARCHAR(20);

DECLARE trip\_cost FLOAT(7,2);

DECLARE date\_end DATETIME;

DECLARE finishedflag INT;

DECLARE costcursor CURSOR FOR

SELECT tr\_cost FROM trip WHERE

tr\_br\_code=br\_trip;

DECLARE dateendcursor CURSOR FOR

SELECT tr\_return FROM trip

WHERE tr\_br\_code=br\_trip;

DECLARE datebegincursor CURSOR FOR

SELECT tr\_departure FROM trip

WHERE tr\_br\_code=br\_trip;

DECLARE rescursor CURSOR FOR

SELECT count(res\_seatnum) FROM reservation

INNER JOIN trip ON res\_tr\_id=tr\_id

WHERE tr\_br\_code=br\_trip group by tr\_id;

DECLARE maxcursor CURSOR FOR

SELECT tr\_maxseats FROM trip

WHERE tr\_br\_code=br\_trip;

DECLARE drcursor CURSOR FOR

SELECT wrk\_name, wrk\_lname FROM worker

INNER JOIN driver ON wrk\_AT=drv\_AT

INNER JOIN trip ON drv\_AT=tr\_drv\_AT

WHERE tr\_br\_code=br\_trip;

DECLARE gucursor CURSOR FOR

SELECT wrk\_name, wrk\_lname FROM worker

INNER JOIN guide ON wrk\_AT=gui\_AT

INNER JOIN trip ON gui\_AT=tr\_gui\_AT

WHERE tr\_br\_code=br\_trip;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN costcursor;

OPEN dateendcursor;

OPEN datebegincursor;

OPEN rescursor;

OPEN maxcursor;

OPEN drcursor;

OPEN gucursor;

REPEAT

FETCH costcursor INTO trip\_cost;

FETCH dateendcursor INTO date\_end;

FETCH datebegincursor INTO date\_begin;

FETCH rescursor INTO res\_seats;

FETCH maxcursor INTO max\_seats;

FETCH drcursor INTO dr\_name, dr\_lname;

FETCH gucursor INTO gu\_name, gu\_lname;

IF (finishedflag=0) THEN

IF (start\_date<=date\_begin AND date\_begin<=end\_date) THEN

SET empty\_seats=max\_seats-res\_seats;

SELECT trip\_cost AS 'trip cost', max\_seats AS 'max seats', res\_seats AS 'seats reserved', empty\_seats AS 'seats available', dr\_name AS 'driver name', dr\_lname AS 'driver lastname', gu\_name AS 'guide name', gu\_lname AS 'guide lastname', date\_begin AS 'departure date', date\_end AS 'return date';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE costcursor;

CLOSE dateendcursor;

CLOSE datebegincursor;

CLOSE rescursor;

CLOSE maxcursor;

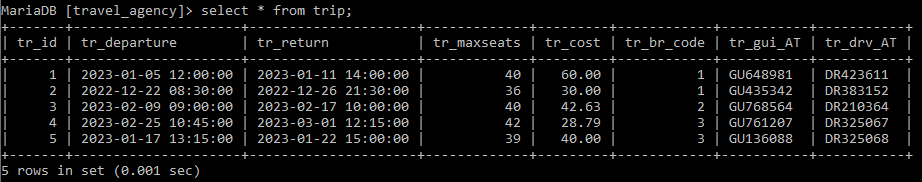
CLOSE drcursor;

CLOSE gucursor;

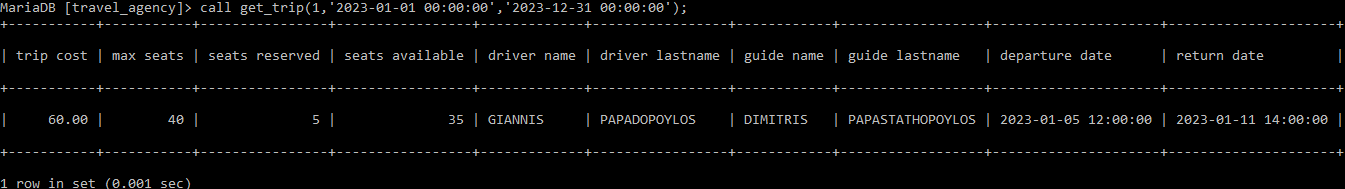
END$

DELIMITER ;

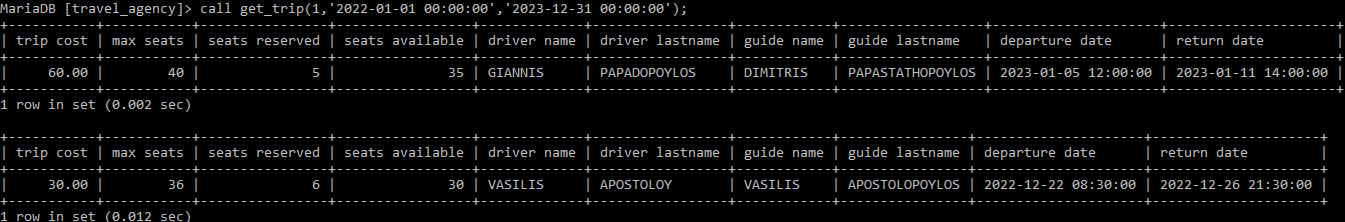
**Πίνακας trip:**



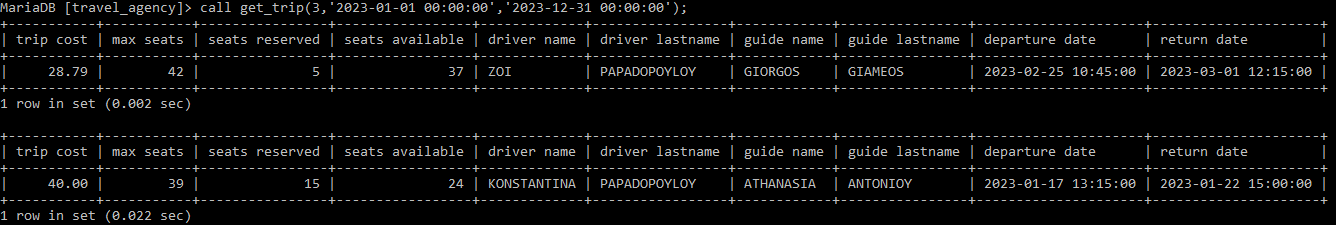
**Παράδειγμα για το branch 1 για ημερομηνίες από 2023-01-01 έως 2023-12-31:**



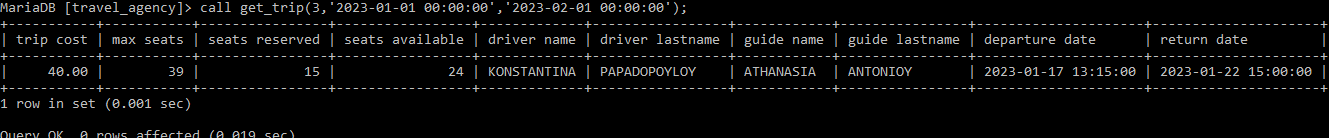
**Παράδειγμα για το branch 1 για ημερομηνίες από 2022-01-01 έως 2023-12-31:**



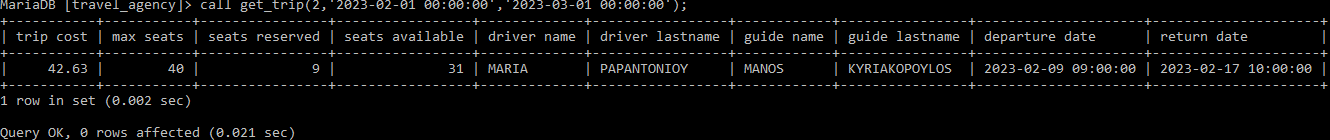
**Παράδειγμα για το branch 3 για ημερομηνίες από 2023-01-01 έως 2023-12-31:**



**Παράδειγμα για το branch 3 για ημερομηνίες από 2023-01-01 έως 2023-02-01:**



**Παράδειγμα για το branch 2 για ημερομηνίες από 2023-02-01 έως 2023-03-01:**



**3.1.3.3**

**Κώδικας procedure για διαγραφή admin:**

DROP PROCEDURE IF EXISTS del\_admin;

DELIMITER $

CREATE PROCEDURE del\_admin(ad\_name VARCHAR(20), ad\_lname VARCHAR(20))

BEGIN

DECLARE manager\_AT CHAR(10);

DECLARE admin\_AT CHAR(10);

SELECT adm\_AT INTO admin\_AT

FROM worker INNER JOIN admin ON wrk\_AT=adm\_AT

WHERE wrk\_name=ad\_name AND wrk\_lname=ad\_lname;

SELECT mng\_adm\_AT INTO manager\_AT

FROM worker INNER JOIN admin ON wrk\_AT=adm\_AT

INNER JOIN manages ON adm\_AT=mng\_adm\_AT

WHERE wrk\_name=ad\_name AND wrk\_lname=ad\_lname;

IF(admin\_AT IS NOT NULL) THEN

IF(manager\_AT IS NOT NULL) THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT = 'Admin given is manager and cannot be deleted';

ELSE

DELETE FROM worker WHERE wrk\_AT=admin\_AT;

END IF;

ELSE

SIGNAL SQLSTATE VALUE '45000'

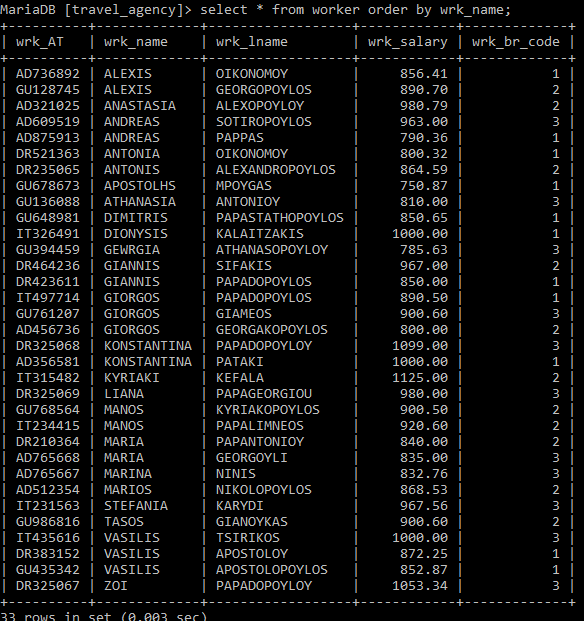
SET MESSAGE\_TEXT = 'Person given is not an admin';

END IF;

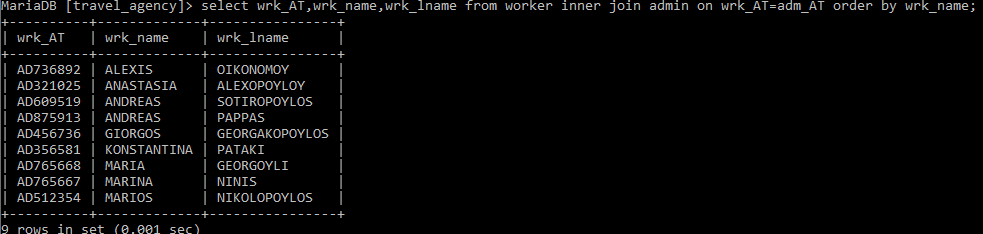
END$

DELIMITER ;

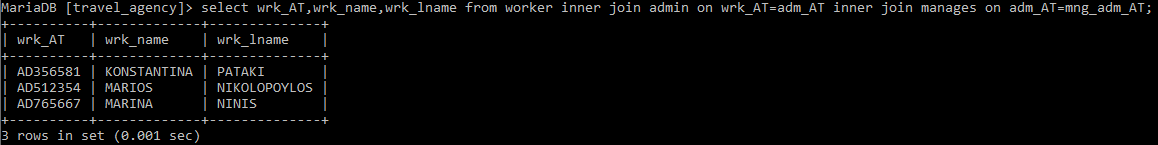
**Πίνακας worker:**



**Όνομα, επώνυμο και ΑΤ των admin:**



**Όνομα, επώνυμο και ΑΤ των admin που είναι manager:**



**Παράδειγμα procedure για άτομο που δεν είναι στην βάση δεδομένων:**



**Παράδειγμα procedure για άτομο που δεν είναι admin(εδώ είναι guide):**

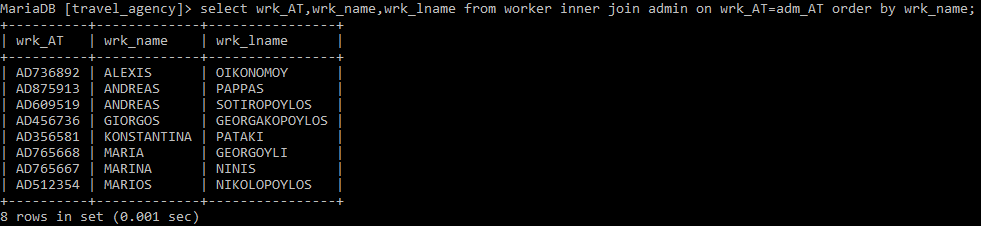
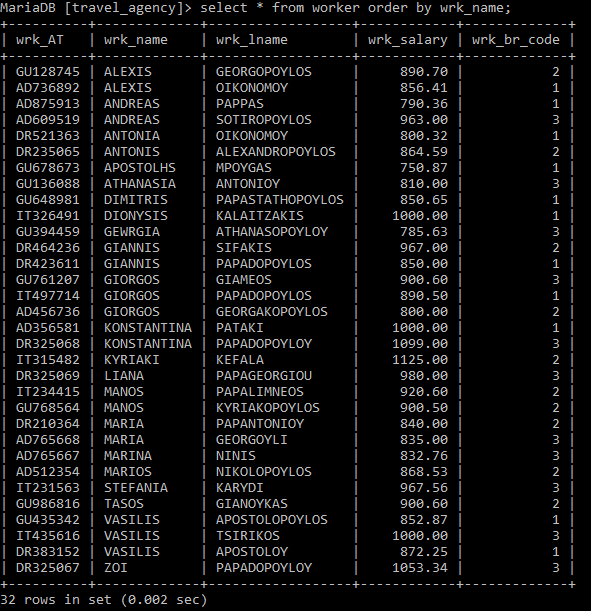


**Παράδειγμα procedure για άτομο που είναι admin και manager:**



**Παράδειγμα procedure για άτομο που είναι admin και δεν είναι manager:**



****

**Όπως φαίνεται η “ANASTASIA ALEXOPOYLOY” έχει διαγραφεί.**

**3.1.3.4**

**α) Κώδικας procedure για επιστροφή ονοματεπωνύμων με βάση την προκαταβολή που πλήρωσαν:**

DROP PROCEDURE IF EXISTS customer\_reservation;

DELIMITER $

CREATE PROCEDURE customer\_reservation(begin\_cost INT,end\_cost INT)

BEGIN

SELECT res\_name AS name,res\_lastname AS lastname

FROM reservation\_offers WHERE adv\_pay>=begin\_cost

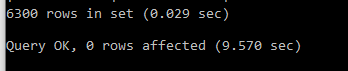
AND adv\_pay<=end\_cost;

END$

DELIMITER ;

**Παράδειγμα χωρίς ευρετήριο:**

**call customer\_reservation(150,165);**



**Παράδειγμα με προσθήκη ευρετηρίου:**

**create index resoffers on reservation\_offers(res\_name,res\_lastname);**

**call customer\_reservation(150,165);**



**β) Κώδικας procedure για επιστροφή ονοματεπωνύμων και προσφορών ταξιδιού με βάση το επώνυμο:**

DROP PROCEDURE IF EXISTS same\_lastname;

DELIMITER $

CREATE PROCEDURE same\_lastname(person\_lname VARCHAR(30))

BEGIN

DECLARE j INT;

DECLARE i INT;

SELECT count(\*) INTO i FROM offers;

SET j=1;

REPEAT

SELECT res\_name AS name,res\_lastname AS lastname,res\_offer\_id,count(\*) AS number\_of\_customers

FROM reservation\_offers

WHERE res\_lastname=person\_lname AND res\_offer\_id=j

GROUP BY res\_name;

SET j=j+1;

UNTIL(j=i+1)

END REPEAT;

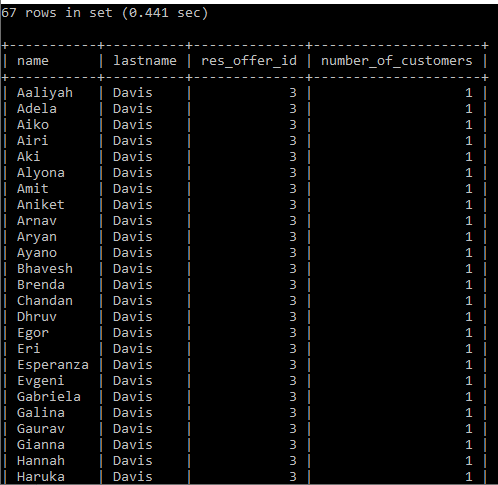
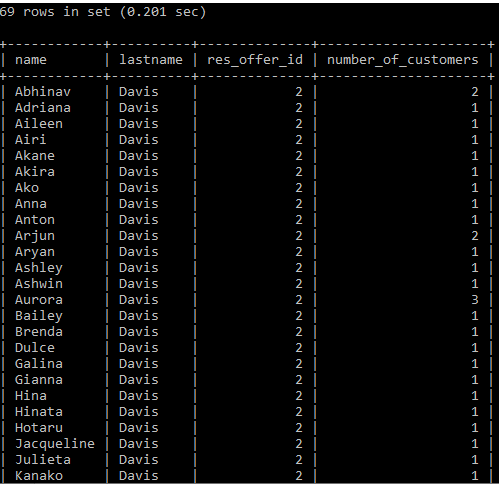
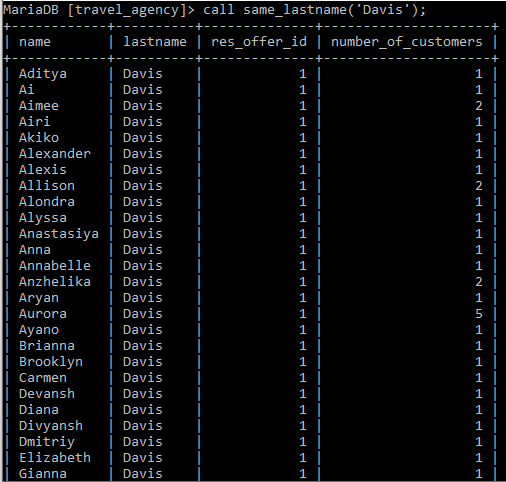
END$

DELIMITER ;

**Παράδειγμα χωρίς ευρετήριο:**

**call same\_lastname('Davis');**

**Τα αποτελέσματα θα βγουν σε διαφορετικούς πίνακες, ένας για το κάθε res\_offer\_id.**



**Υποσημίωση: Οι πίνακες είναι δειγματοληπτικοί.**

**Τελικό αποτέλεσμα:**



**Παράδειγμα με προσθήκη ευρετηρίου:**

**create index resoffers on reservation\_offers(res\_name,res\_lastname);**

**Τελικό αποτέλεσμα:**



**Υποσημείωση: Τα ευρετήρια που χρησιμοποιήσαμε δεν δουλεύουν όπως θα περιμέναμε.**

**3.1.4 Δημιουργία Trigger**

**3.1.4.1**

**Κώδικες trigger για εισαγωγή στον πίνακα log:**

**Insert για τον πίνακα trip:**

DROP TRIGGER IF EXISTS log\_insert\_trip;

DELIMITER $

CREATE TRIGGER log\_insert\_trip

AFTER INSERT ON trip

FOR EACH ROW

BEGIN

IF(new.tr\_br\_code=1) THEN

UPDATE log SET

table\_trip='INSERT',trip\_last\_update=NOW()

WHERE log\_AT='IT497714';

ELSEIF(new.tr\_br\_code=2) THEN

UPDATE log SET

table\_trip='INSERT',trip\_last\_update=NOW()

WHERE log\_AT='IT234415';

ELSEIF(new.tr\_br\_code=3) THEN

UPDATE log SET

table\_trip='INSERT',trip\_last\_update=NOW()

WHERE log\_AT='IT435616';

END IF;

END$

DELIMITER ;

**Update για τον πίνακα trip:**

DROP TRIGGER IF EXISTS log\_update\_trip;

DELIMITER $

CREATE TRIGGER log\_update\_trip

AFTER UPDATE ON trip

FOR EACH ROW

BEGIN

IF(old.tr\_br\_code=1) THEN

UPDATE log SET

table\_trip='UPDATE',trip\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(old.tr\_br\_code=2) THEN

UPDATE log SET

table\_trip='UPDATE',trip\_last\_update=NOW()

WHERE log\_AT='IT315482';

ELSEIF(old.tr\_br\_code=3) THEN

UPDATE log SET

table\_trip='UPDATE',trip\_last\_update=NOW()

WHERE log\_AT='IT231563';

END IF;

END$

DELIMITER ;

**Delete για τον πίνακα trip:**

DROP TRIGGER IF EXISTS log\_delete\_trip;

DELIMITER $

CREATE TRIGGER log\_delete\_trip

AFTER DELETE ON trip

FOR EACH ROW

BEGIN

IF(old.tr\_br\_code=1) THEN

UPDATE log SET

table\_trip='DELETE', trip\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(old.tr\_br\_code=2) THEN

UPDATE log

SET table\_trip='DELETE', trip\_last\_update=NOW()

WHERE log\_AT='IT315482';

ELSEIF(old.tr\_br\_code=3) THEN

UPDATE log SET

table\_trip='DELETE', trip\_last\_update=NOW()

WHERE log\_AT='IT231563';

END IF;

END$

DELIMITER ;

**Insert για τον πίνακα reservation:**

DROP TRIGGER IF EXISTS log\_insert\_reservation;

DELIMITER $

CREATE TRIGGER log\_insert\_reservation

AFTER INSERT ON reservation

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code

FROM trip INNER JOIN reservation

ON tr\_id=res\_tr\_id WHERE res\_tr\_id=new.res\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_reservation='INSERT',reservation\_last\_update=NOW()

WHERE log\_AT='IT497714';

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_reservation='INSERT',reservation\_last\_update=NOW()

WHERE log\_AT='IT234415';

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_reservation='INSERT',reservation\_last\_update=NOW()

WHERE log\_AT='IT435616';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Update για τον πίνακα reservation:**

DROP TRIGGER IF EXISTS log\_update\_reservation;

DELIMITER $

CREATE TRIGGER log\_update\_reservation

AFTER UPDATE ON reservation

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code

FROM trip INNER JOIN reservation

ON tr\_id=res\_tr\_id WHERE res\_tr\_id=new.res\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_reservation='UPDATE',reservation\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_reservation='UPDATE',reservation\_last\_update=NOW()

WHERE log\_AT='IT315482';

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_reservation='UPDATE',reservation\_last\_update=NOW()

WHERE log\_AT='IT231563';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Delete για τον πίνακα reservation:**

DROP TRIGGER IF EXISTS log\_delete\_reservation;

DELIMITER $

CREATE TRIGGER log\_delete\_reservation

BEFORE DELETE ON reservation

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code

FROM trip INNER JOIN reservation

ON tr\_id=res\_tr\_id WHERE res\_tr\_id=old.res\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_reservation='DELETE',reservation\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_reservation='DELETE',reservation\_last\_update=NOW()WHERE log\_AT='IT315482';

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_reservation='DELETE',reservation\_last\_update=NOW()WHERE log\_AT='IT231563';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Insert για τον πίνακα event:**

DROP TRIGGER IF EXISTS log\_insert\_event;

DELIMITER $

CREATE TRIGGER log\_insert\_event

AFTER INSERT ON event

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM event

INNER JOIN trip ON ev\_tr\_id=tr\_id

WHERE ev\_tr\_id=new.ev\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_event='INSERT',event\_last\_update=NOW()

WHERE log\_AT='IT497714';

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_event='INSERT',event\_last\_update=NOW()

WHERE log\_AT='IT234415';

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_event='INSERT',event\_last\_update=NOW()

WHERE log\_AT='IT435616';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Update για τον πίνακα event:**

DROP TRIGGER IF EXISTS log\_update\_event;

DELIMITER $

CREATE TRIGGER log\_update\_event

AFTER UPDATE ON event

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM event

INNER JOIN trip ON ev\_tr\_id=tr\_id

WHERE ev\_tr\_id=old.ev\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET table\_event='UPDATE',event\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(branch\_code=2) THEN

UPDATE log SET table\_event='UPDATE',event\_last\_update=NOW()

WHERE log\_AT='IT315482';

ELSEIF(branch\_code=3) THEN

UPDATE log SET table\_event='UPDATE',event\_last\_update=NOW()

WHERE log\_AT='IT231563';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Delete για τον πίνακα event:**

DROP TRIGGER IF EXISTS log\_delete\_event;

DELIMITER $

CREATE TRIGGER log\_delete\_event

BEFORE DELETE ON event

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM event

INNER JOIN trip ON ev\_tr\_id=tr\_id

WHERE ev\_tr\_id=old.ev\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET table\_event='DELETE',event\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(branch\_code=2) THEN

UPDATE log SET table\_event='DELETE',event\_last\_update=NOW()

WHERE log\_AT='IT315482';

ELSEIF(branch\_code=3) THEN

UPDATE log SET table\_event='DELETE',event\_last\_update=NOW()

WHERE log\_AT='IT231563';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Insert για τον πίνακα travel\_to:**

DROP TRIGGER IF EXISTS log\_insert\_travel\_to;

DELIMITER $

CREATE TRIGGER log\_insert\_travel\_to

AFTER INSERT ON travel\_to

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM travel\_to

INNER JOIN trip ON to\_tr\_id=tr\_id

WHERE to\_tr\_id=new.to\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_travel\_to='INSERT',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT497714';

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_travel\_to='INSERT',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT234415';

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_travel\_to='INSERT',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT435616';

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Update για τον πίνακα travel\_to:**

DROP TRIGGER IF EXISTS log\_update\_travel\_to;

DELIMITER $

CREATE TRIGGER log\_update\_travel\_to

AFTER UPDATE ON travel\_to

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM travel\_to

INNER JOIN trip ON to\_tr\_id=tr\_id

WHERE to\_tr\_id=old.to\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF (finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_travel\_to='UPDATE',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_travel\_to='UPDATE',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT315482';

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_travel\_to='UPDATE',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT231563';

END IF;

END IF;

UNTIL (finishedflag=1) END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Delete για τον πίνακα travel\_to:**

DROP TRIGGER IF EXISTS log\_delete\_travel\_to;

DELIMITER $

CREATE TRIGGER log\_delete\_travel\_to

BEFORE DELETE ON travel\_to

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM travel\_to

INNER JOIN trip ON to\_tr\_id=tr\_id

WHERE to\_tr\_id=old.to\_tr\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF(finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_travel\_to='DELETE',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT326491';

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_travel\_to='DELETE',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT315482';

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_travel\_to='DELETE',travel\_to\_last\_update=NOW()

WHERE log\_AT='IT231563';

END IF;

END IF;

UNTIL (finishedflag=1) END REPEAT;

CLOSE brcodecursor;

END$

DELIMITER ;

**Insert για τον πίνακα destination:**

DROP TRIGGER IF EXISTS log\_insert\_destination;

DELIMITER $

CREATE TRIGGER log\_insert\_destination

AFTER INSERT ON destination

FOR EACH ROW

BEGIN

UPDATE log SET

table\_destination='INSERT',destination\_last\_update=NOW()

WHERE log\_AT='IT497714';

END$

DELIMITER ;

**Update για τον πίνακα destination:**

DROP TRIGGER IF EXISTS log\_update\_destination;

DELIMITER $

CREATE TRIGGER log\_update\_destination

AFTER UPDATE ON destination

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE s INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM destination

INNER JOIN travel\_to ON dst\_id=to\_dst\_id

INNER JOIN trip ON to\_tr\_id=tr\_id

WHERE dst\_id=old.dst\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET s=0;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF(finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_destination='UPDATE',destination\_last\_update=NOW()

WHERE log\_AT='IT326491';

SET s=1;

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_destination='UPDATE',destination\_last\_update=NOW()

WHERE log\_AT='IT315482';

SET s=1;

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_destination='UPDATE',destination\_last\_update=NOW()

WHERE log\_AT='IT231563';

SET s=1;

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

IF(s=0) THEN

UPDATE log SET table\_destination='UPDATE',destination\_last\_update=NOW()

WHERE log\_AT='IT326491';

END IF;

END$

DELIMITER ;

**Delete για τον πίνακα destination:**

DROP TRIGGER IF EXISTS log\_delete\_destination;

DELIMITER $

CREATE TRIGGER log\_delete\_destination

BEFORE DELETE ON destination

FOR EACH ROW

BEGIN

DECLARE branch\_code INT;

DECLARE finishedflag INT;

DECLARE s INT;

DECLARE brcodecursor CURSOR FOR

SELECT tr\_br\_code FROM destination

INNER JOIN travel\_to ON dst\_id=to\_dst\_id

INNER JOIN trip ON to\_tr\_id=tr\_id

WHERE dst\_id=old.dst\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

SET s=0;

SET finishedflag=0;

OPEN brcodecursor;

REPEAT

FETCH brcodecursor INTO branch\_code;

IF(finishedflag=0) THEN

IF(branch\_code=1) THEN

UPDATE log SET

table\_destination='DELETE',destination\_last\_update=NOW()

WHERE log\_AT='IT326491';

SET s=1;

ELSEIF(branch\_code=2) THEN

UPDATE log SET

table\_destination='DELETE',destination\_last\_update=NOW()

WHERE log\_AT='IT315482';

SET s=1;

ELSEIF(branch\_code=3) THEN

UPDATE log SET

table\_destination='DELETE',destination\_last\_update=NOW()

WHERE log\_AT='IT231563';

SET s=1;

END IF;

END IF;

UNTIL (finishedflag=1)

END REPEAT;

CLOSE brcodecursor;

IF(s=0) THEN

UPDATE log SET table\_destination='UPDATE',destination\_last\_update=NOW()

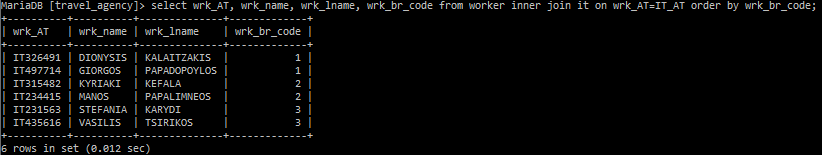
WHERE log\_AT='IT326491';

END IF;

END$

DELIMITER ;

**Παραδοχή 1: Σε κάθε παράρτημα υπάρχουν δύο υπεύθυνοι υπολογιστών. Οπότε για να καθορίσουμε το ποιος κάνει την κάθε ενέργεια τους χωρίσαμε σε δύο κατηγορίες. Ο ένας κάνει τις εισαγωγές και ο άλλος κάνει ελέγχους. Δηλαδή ο ένας θα ασχολείται με τα insert και ο άλλος με τα update και delete.**

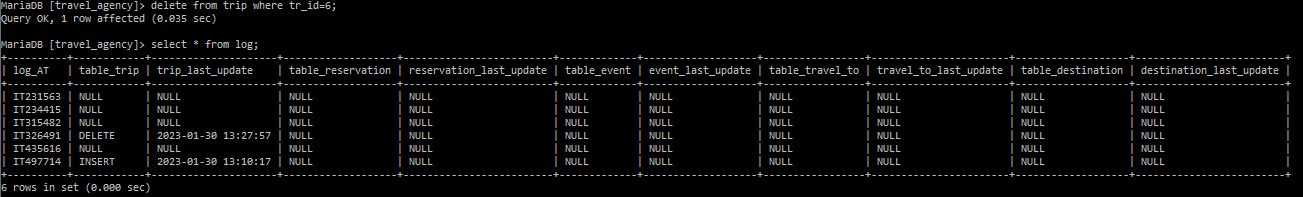
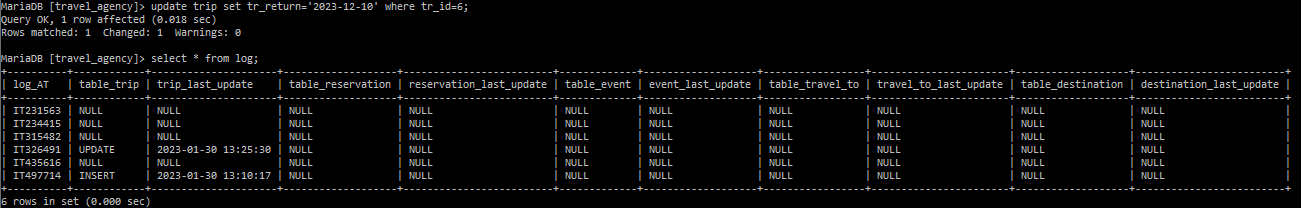
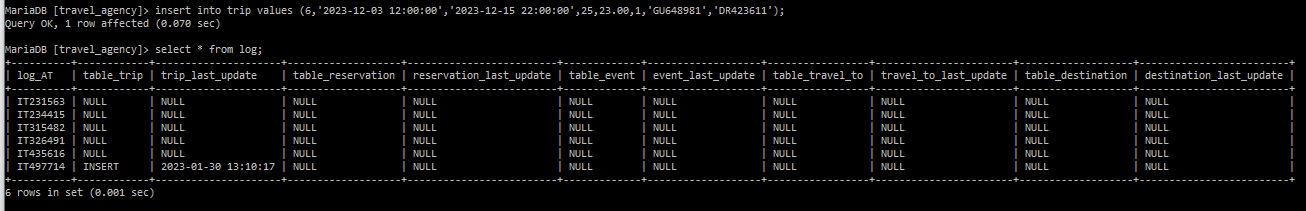


**Οι IT που θα ασχολούνται με τα insert είναι οι “GIORGOS PAPADOPOYLOS IT497714”(branch 1), “MANOS PAPLIMNEOS IT234415”(branch 2), “VASILIS TSIRIKOS IT435616”(branch 3).**

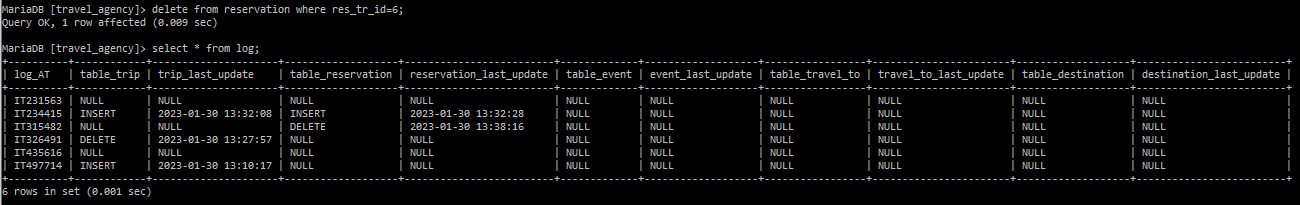
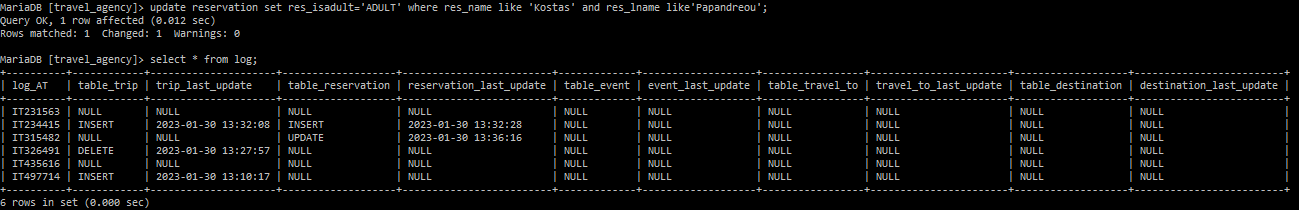
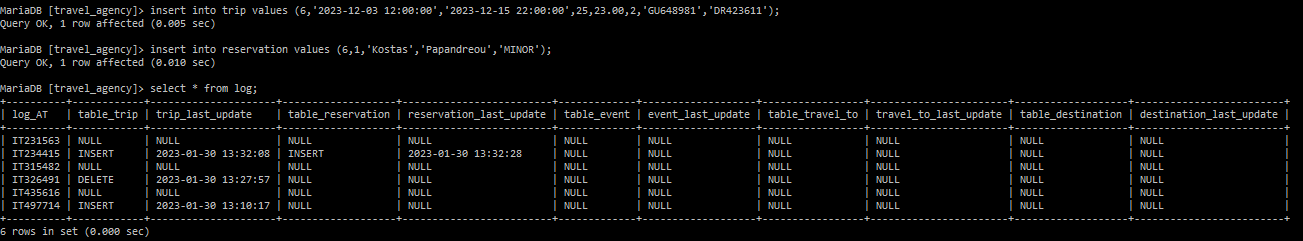
**Οι IT που θα ασχολούνται με τα update και delete είναι οι “DIONYSIS KALAITZAKIS IT326491”(branch 1), “KYRIAKI KEFALA IT315482”(branch 2), “STEFANIA KARYDI IT231563”(branch 3).**

**Παραδοχή 2: Ο πίνακας destination δεν συνδέεται με τον πίνακα trip αν δεν υπάρχουν δεδομένα στον πίνακα travel\_to. Οπότε κάναμε την παραδοχή ότι κάθε καινούργιο destination θα εισάγεται από τον IT του branch 1 και οι αντίστοιχες αναβαθμίσεις ή διαγραφές θα γίνονται από το branch που ανέλαβε τελικά αυτό το ταξίδι. Επίσης αν κάποιο destination χρειάζεται αναβάθμιση ή διαγραφή και δεν αντιστοιχίζεται με κάποιο travel\_to, η ενέργεια θα γίνεται από τον IT του branch 1.**

**Παραδείγματα για τον πίνακα trip:**

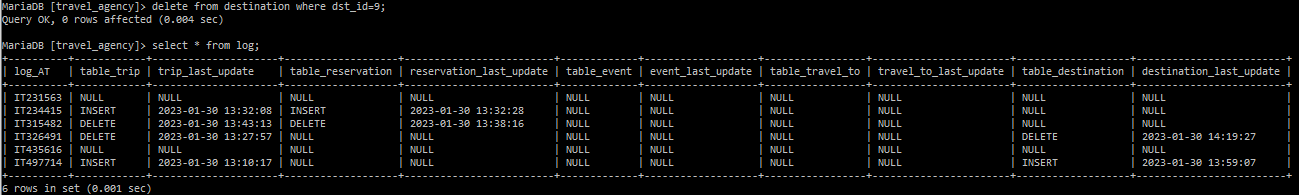
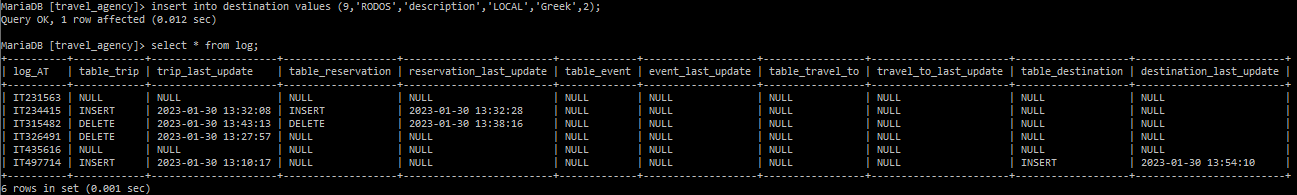


**Παραδείγματα για τον πίνακα reservation:**

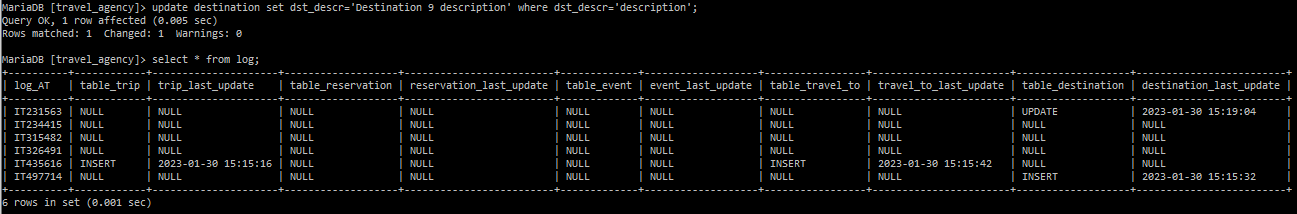
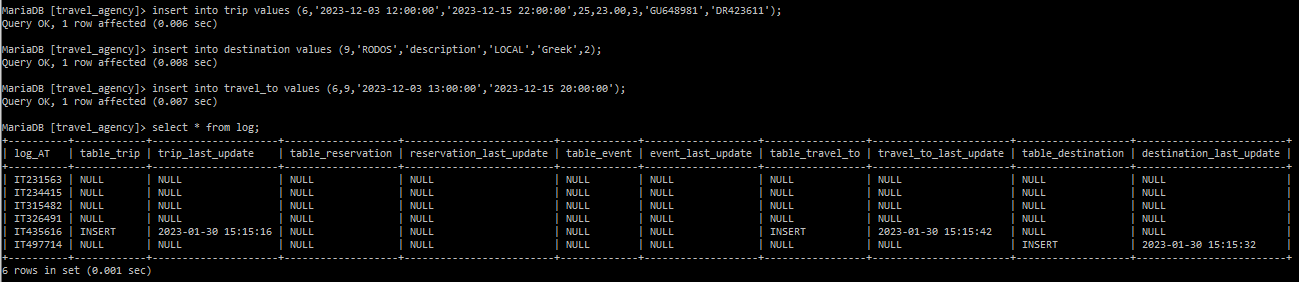


**Παρόμοια θα είναι τα αποτελέσματα και για τις στήλες που αφορούν τους πίνακες event και travel\_to.**

**Παραδείγματα για τον πίνακα destination χωρίς σύνδεση με τον πίνακα travel\_to:**



**Παραδείγματα για τον πίνακα destination με σύνδεση με τον πίνακα travel\_to:**



**3.1.4.2**

**Κώδικας για αποτροπή διαγραφής ημερομηνίας αναχώρησης και επιστροφής και κόστους:**

DROP TRIGGER IF EXISTS prevent\_trip\_update;

DELIMITER $

CREATE TRIGGER prevent\_trip\_update

BEFORE UPDATE ON trip

FOR EACH ROW

BEGIN

DECLARE tr\_res INT;

SELECT count(\*) INTO tr\_res

FROM reservation INNER JOIN trip

ON res\_tr\_id=tr\_id WHERE tr\_id=old.tr\_id;

IF(old.tr\_departure!=new.tr\_departure) THEN

IF(tr\_res>0) THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT ='Cannot change departure because there are already reservations for this trip.';

END IF;

END IF;

IF(old.tr\_return!=new.tr\_return) THEN

IF(tr\_res>0) THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT ='Cannot change return because there are already reservations for this trip.';

END IF;

END IF;

IF(old.tr\_cost!=new.tr\_cost) THEN

IF(tr\_res>0) THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT ='Cannot change cost because there are already reservations for this trip.';

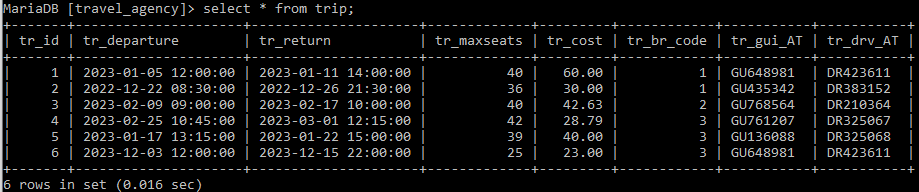
END IF;

END IF;

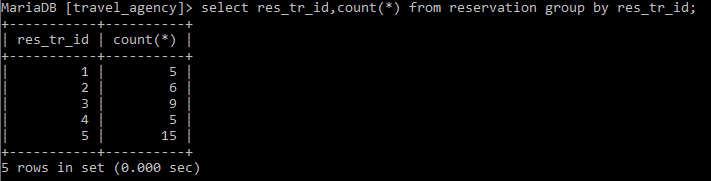
END$

DELIMITER ;

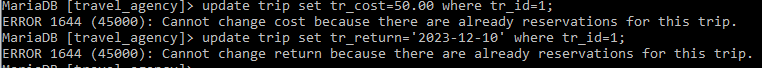
**Πίνακας trip:**



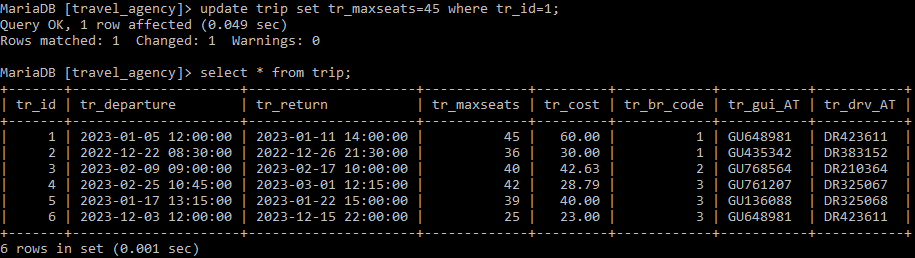
**Αριθμός κρατήσεων ανά ταξίδι:**



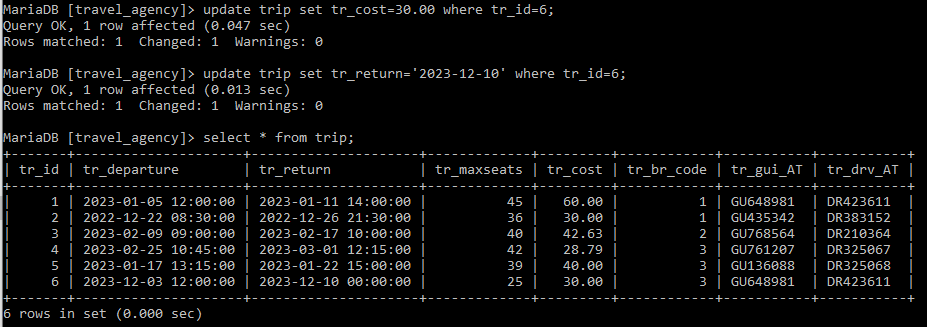
**Παράδειγμα για αλλαγή κόστους ή ημερομηνίας επιστροφής σε ταξίδι που έχει κρατήσεις:**



**Παράδειγμα για αλλαγή μεγίστων θέσεων σε ταξίδι που έχει κρατήσεις:**



**Παράδειγμα για αλλαγή κόστους ή ημερομηνίας επιστροφής σε ταξίδι που δεν έχει κρατήσεις:**



**3.1.4.3**

**Κώδικας για αποτροπή μείωσης μισθού:**

DROP TRIGGER IF EXISTS prevent\_salary\_decrease;

DELIMITER $

CREATE TRIGGER prevent\_salary\_decrease

BEFORE UPDATE ON worker

FOR EACH ROW

BEGIN

IF(new.wrk\_salary<old.wrk\_salary) THEN

SIGNAL SQLSTATE VALUE '45000'

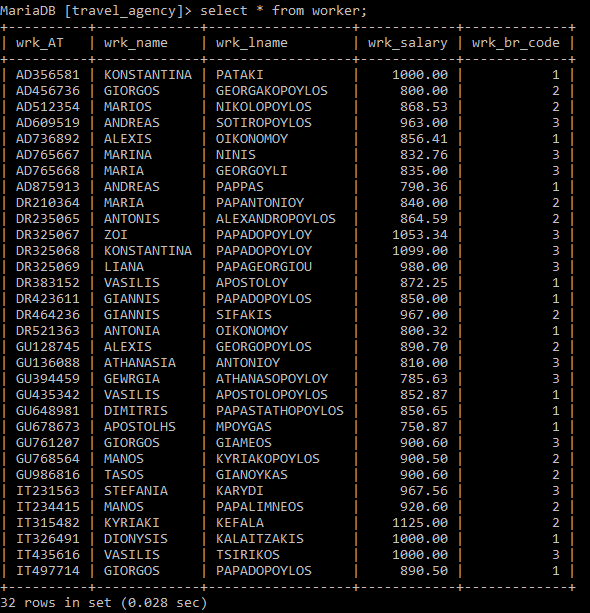
SET MESSAGE\_TEXT ='Cannot lower salary of worker.';

END IF;

END$

DELIMITER ;

**Πίνακας worker:**



**Παράδειγμα για προσπάθεια μείωσης μισθού:**



**Παράδειγμα για προσπάθεια αύξησης μισθού:**

