

# PROJECT ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

## TRAVEL\_AGENCY

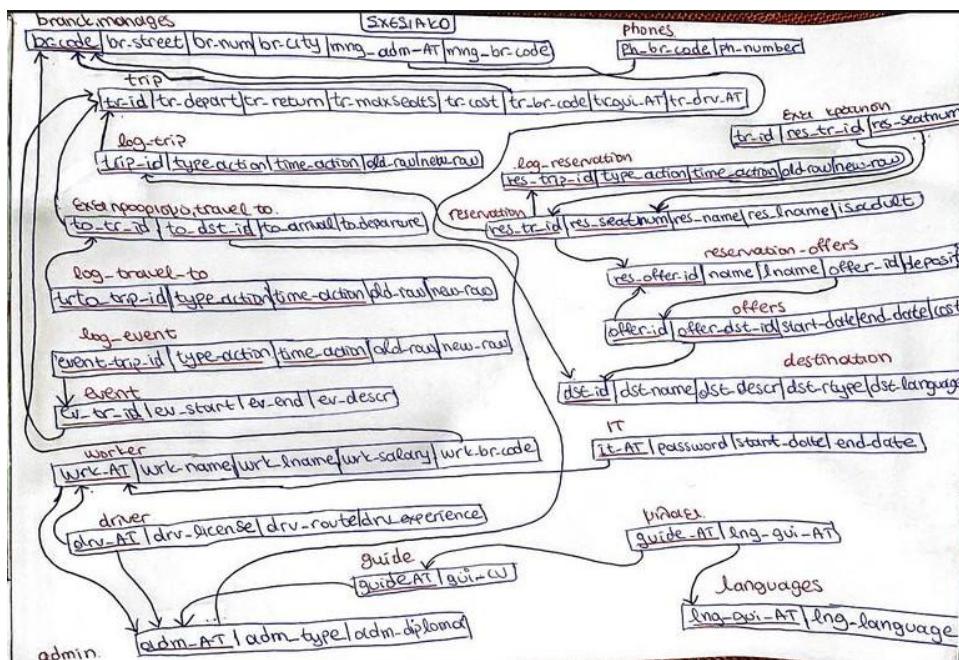
Μαριάνθη Θώδη ΑΜ:1084576

Ελπίδα Κόκκαλη ΑΜ: 1084648

3ο έτος

### Α' μέρος :

#### Κεφάλαιο 1



- **table it** : Είναι ένας πίνακας που θα περιλαμβάνει όλους τους διαχειριστές της βάσης δεδομένων travel\_agency. Περιέχει το αφμ και τον κωδικό πρόσβασης κάθε διαχειριστή που με αυτά θα μπορεί να συνδεθεί ο εκάστοτε στην διεπαφή GUI . Επίσης περιέχει την ημερομηνία έναρξης ,δηλαδή πότε ξεκίνησε ένας worker να είναι διαχειριστής της βάσης καθώς και την ημερομηνία που σταμάτησε να επιτελεί έργα διαχειριστή .Αν αυτό το πεδίο είναι NULL ,αυτό σημαίνει ότι ακόμα επιτελεί έργο διαχειριστή.

- **table offers** : Είναι ένας πίνακας που περιλαμβάνει κάποιες ειδικές προσφορές ,που σχετίζονται με συγκεκριμένο προορισμό ,που μπορούν να πραγματοποιηθούν κατόπιν κράτησης , σε συγκεκριμένο χρονικό διάστημα .Το offer\_id αναφέρεται σε ένα συγκεκριμένο προορισμό από τον πίνακα destination.
- **table reservation offers**: Είναι ένας πίνακας που περιλαμβάνει τις κρατήσεις για κάθε προσφορά για έναν συγκεκριμένο προορισμό και περιέχει τα στοιχεία κάθε πελάτη για την κράτηση.
- **table log\_trip/ log\_event/log\_reservation/log\_travel\_to** : Είναι πίνακες που όταν ένα από τα αντίστοιχα triggers(update/insert/delete) ενεργοποιηθεί ,θα καταγράφουν τις ενέργειες που πραγματοποιήθηκαν από τον διαχειριστή .

— **Λεπτομέρεις:**

1.

import random //κώδικας στην python για να κάνω τις 60.000 εγγραφές ,όπου letssee.txt το αρχείο με τα 1000 ονόματα

```
file=open('letssee.txt','r')
data=file.readlines()
firstnames=[]
lastnames=[]

i=0

for line in data:
    names=line.strip('\n').split(',')
    for string in names:
        x=random.randint(50, 200)
        name=string.split('\t')[:-1]
        for c in name:
```

```

firstname=string.split('\t')[0]
firstnames.append(firstname)
lastname=string.split('\t')[-1]
lastnames.append(lastname)
break

print(f'INSERT INTO reservation_offers(name,lname,offer_id,deposit) VALUES
("'{firstnames[i]}","'{lastnames[i]}",1or2or3****,{x} );')

i=i+1

```

\*\*\*\*\*1 για τις πρωτες 20.000 εγγραφες ,2 για τις επομενες 20.000 και 3 για τις επομενες 20.000

2. Στον πίνακα admin ,οι 20 πρώτες εγγραφές είναι πρόσθετες και αναφέρονται σε διοικητικούς υπαλλήλους που έχουν αναλάβει την διοίκηση ενός υποκαταστήματος.

---

## *Κεφάλαιο 2*

### **STORE PROCEDURES:**

#### **3.1.3.1**

DELIMITER \$

```

CREATE PROCEDURE new_driver(IN driver_name VARCHAR(20), IN driver_lname
VARCHAR(20), IN driver_salary FLOAT(7,2),
IN driver_AT VARCHAR(10),IN driver_license ENUM('A','B','C','D'),
IN driver_route ENUM('LOCAL', 'ABROAD'),IN driver_experience TINYINT)
BEGIN
DECLARE branch_id INT;
DECLARE sum_drivers INT ;

```

```

SELECT br.br_code,COUNT(drv.drv_AT)
INTO branch_id,sum_drivers

FROM branch AS br JOIN worker AS wrk ON br.br_code=wrk.wrk_br_code
JOIN driver AS drv ON wrk.wrk_AT=drv.drv_AT
GROUP BY br.br_code
ORDER BY COUNT(drv.drv_AT)

LIMIT 1;

```

```

INSERT INTO worker VALUES(driver_AT,driver_name,driver_lname,driver_salary,branch_id);

INSERT INTO driver VALUES(driver_AT,driver_license,driver_route,driver_experience);

END$
```

DELIMITER;

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```

1 • SELECT br.br_code,COUNT(drv.drv_AT)
2   FROM branch AS br JOIN worker AS wrk ON br.br_code=wrk.wrk_br_code
3     JOIN driver AS drv ON wrk.wrk_AT=drv.drv_AT
4   GROUP BY br.br_code
5   ORDER BY COUNT(drv.drv_AT)
6
7

```

The results grid displays the following data:

br_code	COUNT(drv.drv_AT)
10	1
9	1
8	1
7	1
6	1
5	1
4	1
3	1
2	1
1	1
11	1
12	1
13	1
14	1
17	1
18	1
19	1
20	1
16	2

//Επιλέγει το πρώτο υποκατάστημα με τους λιγότερους οδηγούς

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1 SQL File 3\*

```

1 • SELECT br.br_code,COUNT(drv.drv_AT)
2 FROM branch AS br JOIN worker AS wrk ON br.br_code=wrk.wrk_br_code
3 JOIN driver AS drv ON wrk.wrk_AT=drv.drv_AT
4 GROUP BY br.br_code
5 ORDER BY COUNT(drv.drv_AT)
6 LIMIT 1;
7

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

br_code	COUNT(drv.drv_AT)
10	1

Administration Schemas

Information: Schema: travel\_agency

//Έγινε η εισαγωγή

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Query 1 SQL File 3\*

```

1 • call new_driver('giannhs','giannaks',65,'1111111','A','LOCAL',3);
2
3 • SELECT br.br_code,COUNT(drv.drv_AT)
4 FROM branch AS br JOIN worker AS wrk ON br.br_code=wrk.wrk_br_code
5 JOIN driver AS drv ON wrk.wrk_AT=drv.drv_AT
6 GROUP BY br.br_code
7 ORDER BY COUNT(drv.drv_AT);

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

br_code	COUNT(drv.drv_AT)
9	1
8	1
7	1
6	1
5	1
4	1
3	1
2	1
1	1
11	1
12	1
13	1
14	1
17	1
18	1
19	1
20	1
10	2
16	2

Administration Schemas

Information: Schema: travel\_agency

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help' are visible. Below the menu is a toolbar with icons for file operations like Open, Save, and Print. The 'Navigator' pane on the left lists various database objects under the 'SCHEMAS' section, including 'language', 'log\_event', 'log\_reservation', 'log\_travel\_to', 'log\_trip', 'manages', 'offers', 'phone', 'reservation', 'reservation\_offers', 'travel\_to', 'trip', and 'worker'. The 'Administration' tab is selected. The main area contains a 'Query 1' tab with the following SQL code:

```

1 •  call new_driver('giannhs','giannakhs',65,'1111111','A','LOCAL',3);
2 •  select* from driver;
3

```

Below the code is a 'Result Grid' showing the output of the second query:

	drv_AT	drv_license	drv_route	drv_experience
1001001	B	ABROAD	4	
1001010	C	LOCAL	1	
1001011	D	ABROAD	4	
1111111	A	LOCAL	3	
*	HULL	NULL	NULL	

### 3.1.3.3

DELIMITER \$

```
CREATE PROCEDURE delete_worker( IN name_worker VARCHAR(20), IN lname_worker
VARCHAR(20))
```

BEGIN

```
DECLARE admtype VARCHAR(20);
```

```
DECLARE keyadm VARCHAR(10);
```

```
DECLARE result VARCHAR(255);
```

```
SELECT adm.adm_type,adm.adm_AT
```

```
INTO admtype,keyadm
```

```
FROM worker AS wrk JOIN admin AS adm ON wrk.wrk_AT=adm.adm_AT
```

```
WHERE wrk.wrk_name=name_worker AND wrk.wrk_lname=lname_worker;
```

```
SELECT IF(EXISTS (SELECT adm.adm_AT
```

```
FROM admin AS adm JOIN manages AS mn ON adm.adm_AT=mn.mng_adm_AT
```

```
WHERE mn.mng_adm_AT=keyadm),'Ο SUGKEKRIMENOS UPALLHLOS EINAI  
DIEUTHUNTHS','Ο SUGKEKRIMENOS UPALLHLOS DEN EINAI DIEUTHUNTHS') INTO result;
```

```
IF (admtpe='ADMINISTRATIVE' AND result='Ο SUGKEKRIMENOS UPALLHLOS EINAI  
DIEUTHUNTHS') THEN
```

```
SELECT 'DEN MPOREIS NA DIAGRASEIS DIEUTHUNTH';
```

```
ELSE
```

```
DELETE FROM worker WHERE wrk_AT=keyadm ;
```

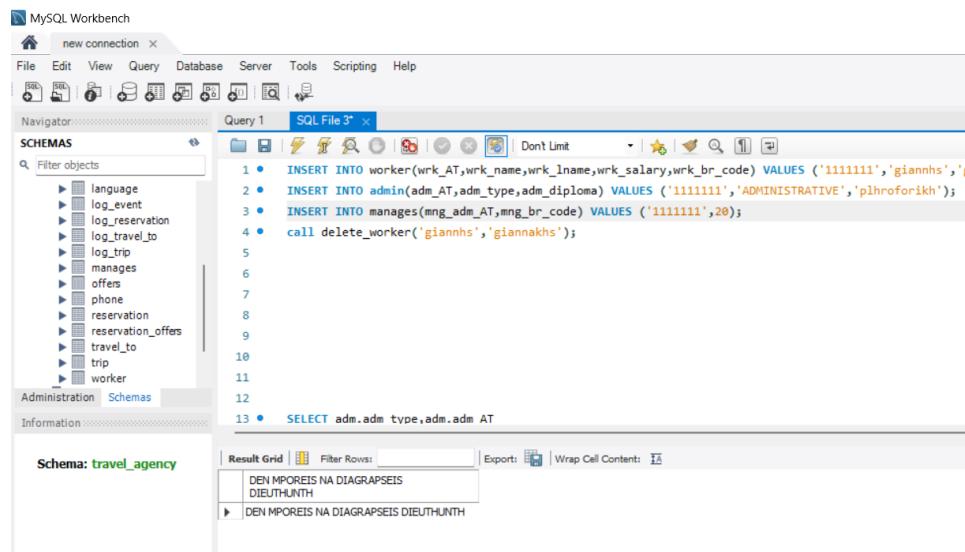
```
SELECT 'DIAGRAFHKE O UPALLHLOS' ;
```

```
END IF;
```

```
END$
```

```
DELIMITER;
```

```
//Εστω ότι προσθέτω έναν υπάλληλο που είναι και διοικητικός και είναι  
διευθυντής του υποκαταστήματος με κωδικό br_code=20. Όταν πάω να τον  
διαγράψω ,εμφανίζεται κατάλληλο μήνυμα.
```



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, which includes tables like language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, offers, phone, reservation, reservation\_offers, travel\_to, trip, and worker. The main area contains a 'Query 1' tab with the following SQL code:

```
1 • INSERT INTO worker(wrk_AT,wrk_name,wrk_lname,wrk_salary,wrk_br_code) VALUES ('1111111','giannhs','g  
2 • INSERT INTO admin(admin_AT,adm_type,adm_diploma) VALUES ('1111111','ADMINISTRATIVE','plhroforikh');  
3 • INSERT INTO manages(mng_adm_AT,mng_br_code) VALUES ('1111111',20);  
4 • call delete_worker('giannhs','giannakhs');  
5  
6  
7  
8  
9  
10  
11  
12  
13 • SELECT adm.adm_type,adm.adm_AT
```

The 'Result Grid' tab shows the output of the last query:

DEN MPOREIS NA DIAGRASEIS DIEUTHUNTH
DEN MPOREIS NA DIAGRASEIS DIEUTHUNTH

**//Εστω ότι προσθέτω απλά μία διοικητική υπάλληλο.**

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

language log\_event log\_reservation log\_travel\_to log\_trip manages offers phone reservation reservation\_offers travel\_to trip worker

Administration Schemas

Information

Schema: travel\_agency

Query 1 SQL File 3\*

```

1 • INSERT INTO worker(wrk_AT,wrk_name,wrk_lname,wrk_salary,wrk_br_code) VALUES ('1111110','sonia','sona',45,3);
2 • INSERT INTO admin(adm_AT,adm_type,adm_diploma) VALUES ('1111110','ADMINISTRATIVE','plhroforikh');
3 • select * from admin;
4
5
6
7
8
9
10
11
12
13

```

Result Grid | Filter Rows: | Edits: | Export/Import: | Wrap Cell Content: |

adm_AT	adm_type	adm_diploma
1001110	LOGISTIC	managment
1001111	LOGISTIC	management
1010000	LOGISTIC	management
1111110	ADMINISTRATIVE	phroforikh

//διαγράφτηκε η υπάλληλος ,εφόσον δεν ήταν διευθύντρια

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

language log\_event log\_reservation log\_travel\_to log\_trip manages offers phone reservation reservation\_offers travel\_to trip worker

Administration Schemas

Information

Schema: travel\_agency

Query 1 SQL File 3\*

```

1 • INSERT INTO worker(wrk_AT,wrk_name,wrk_lname,wrk_salary,wrk_br_code) VALUES ('1111110','sonia','sona',45,3);
2 • INSERT INTO admin(adm_AT,adm_type,adm_diploma) VALUES ('1111110','ADMINISTRATIVE','plhroforikh');
3 • call delete_worker('sonia','sona');
4
5
6
7
8
9
10
11
12
13

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

DIAGRAFTHKE O UPALLHLOS

DIAGRAFTHKE O UPALLHLOS

//Επαλήθευση

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

language log\_event log\_reservation log\_travel\_to log\_trip manages offers phone reservation reservation\_offers travel\_to trip worker

Administration Schemas

Information

Schema: travel\_agency

Query 1 SQL File 3\*

```

1
2
3 • select * from admin;
4
5
6
7
8
9
10
11
12
13

```

Result Grid | Filter Rows: | Edits: | Export/Import: | Wrap Cell Content: |

adm_AT	adm_type	adm_diploma
1001101	LOGISTIC	team leader
1001110	LOGISTIC	managment
1001111	LOGISTIC	management
1010000	LOGISTIC	managment
HULL	HULL	HULL

### 3.1.3.2

//Ο χρήστης ψάχνει να βρει διαθέσιμα ταξίδια στις ημερομηνίες που βάζουμε σαν ορίσματα καθώς και τις υπόλοιπες πληροφορίες για τα ταξίδια αυτά.

//Εκανα μία πρόσθετη εγγραφή στο υποκατάστημα με br\_code=1 ώστε ένα υποκατάστημα να διοργανώνει 2 ταξίδια

```
INSERT INTO destination(dst_id,dst_name,dst_descr,dst_rtype,dst_language,dst_location)
VALUES(20,'hawaii','xalarwtikes diakopes','ABROAD','agglika',20);
```

```
INSERT INTO
trip(tr_departure,tr_return,tr_maxseats,tr_cost,tr_br_code,tr_gui_AT,tr_drv_AT)
VALUES('2023-05-05 09:00:00','2023-10-05 18:00:00',10,500,1,'0100100','0111000');
```

```
INSERT INTO travel_to(to_tr_id,to_dst_id,to_arrival,to_departure)
VALUES (21,20,'2023-01-01 17:00:00','2023-01-05 19:00:00');
```

DELIMITER \$

```
CREATE PROCEDURE check_trip( IN brcode INT , IN date1 DATE, IN date2 DATE)
```

```
BEGIN
```

```
DECLARE flag INT;
```

```
DECLARE trid INT;
```

```
DECLARE cr CURSOR FOR
```

```
SELECT tr.tr_id
```

```
FROM branch AS br JOIN trip AS tr ON br.br_code=tr.tr_br_code
```

```
WHERE br_code=brcode AND DATE(tr.tr_departure) BETWEEN date1 AND date2;
```

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag=1;
```

```
OPEN cr;
```

```

SET flag=0;

WHILE(flag=0) DO

  FETCH cr INTO trid;

  SELECT brcode AS kwdikos_upokatasthamatos,tr.tr_id AS kwdikos_taksidiou,tr.tr_cost AS
kostos_taksidiou ,tr.tr_maxseats AS megistes_theseis,tr.tr_departure AS hm_anaxwrishs
,tr.tr_return AS hm_epistrofhs,(tr.tr_maxseats-COUNT(res.res_tr_id)) AS kenesttheseis

  FROM trip AS tr JOIN reservation AS res ON tr.tr_id=res.res_tr_id

  WHERE tr.tr_id=trid

  GROUP BY res.res_tr_id;

  DO SLEEP(7);

  SELECT brcode AS kwdikos_upokatasthamatos,tr.tr_id AS
kwdikos_taksidiou,CONCAT(wrk_name ,'',wrk.wrk_lname) AS onomatepwnumo_odhgou

  FROM trip AS tr JOIN driver AS drv ON tr.tr_drv_AT=drv.drv_AT JOIN worker AS wrk ON
drv.drv_AT=wrk.wrk_AT

  WHERE tr.tr_id=trid;

  DO SLEEP(7);

  SELECT brcode AS kwdikos_upokatasthamatos,tr.tr_id AS
kwdikos_taksidiou,CONCAT(wrk.wrk_name,'',wrk.wrk_lname) AS
onomatepwnumo_ksenagou

  FROM trip AS tr JOIN guide AS gui ON tr.tr_gui_AT=gui.gui_AT JOIN worker AS wrk ON
gui.gui_AT=wrk.wrk_AT

  WHERE tr.tr_id=trid;

  DO SLEEP(7);

  END WHILE;

  CLOSE cr;

END$


DELIMITER;

//tr_id=1 ,tr_id=27 ανήκουν στο υποκατάστημα με br_code=1
//ο cursor εκτυπώνει τα αποτελέσματα και έχω βάλει τον ίδιο οδηγό και
ξεναγό και στα δύο ταξίδια

```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

SCHEMAS Filter objects

- language
- log\_event
- log\_reservation
- log\_travel\_to
- log\_trip
- manages
- offers
- phone
- reservation
- reservation\_offers
- travel\_to
- trip
- worker

Administration Schemas

Information Schema: travel\_agency

Query 1 SQL File 3

```
1
2 • select * from trip;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

tr_id	tr_departure	tr_return	tr_maxseats	tr_cost	tr_br_code	tr_gua_AT	tr_drv_AT
1	2023-01-05 09:00:00	2023-01-05 18:00:00	10	500.00	1	0100100	0111000
2	2023-01-02 09:00:00	2023-01-06 18:00:00	30	2000.00	2	0100101	0111001
3	2023-01-03 07:00:00	2023-01-07 16:00:00	100	500.00	3	0100110	0111010
4	2023-01-04 12:00:00	2023-01-08 19:00:00	10	200.00	4	0100111	0111011
5	2023-01-05 07:00:00	2023-01-09 20:00:00	40	900.00	5	0101000	0111100
6	2023-01-05 08:00:00	2023-01-07 19:00:00	5	1000.00	6	0101001	0111101
7	2023-02-08 12:00:00	2023-02-13 15:00:00	10	300.00	7	0101010	0111110
8	2023-04-19 12:00:00	2023-04-23 16:00:00	10	300.00	8	0101011	0111111
9	2023-05-19 12:00:00	2023-05-24 16:00:00	100	900.00	9	0101100	1000000
10	2023-01-23 12:00:00	2023-01-28 19:00:00	10	10000.00	10	0101101	1000001
11	2023-02-23 09:00:00	2023-02-28 18:00:00	25	600.00	11	0101110	1000010
12	2023-02-01 10:00:00	2023-02-05 17:00:00	35	700.00	12	0101111	1000011
13	2023-06-23 11:00:00	2023-06-30 14:00:00	45	800.00	11	0110000	1000100
14	2023-03-07 12:00:00	2023-03-18 15:00:00	10	800.00	12	0110001	1000101
15	2023-01-23 07:00:00	2023-01-29 18:00:00	12	700.00	13	0110010	1000110
16	2023-05-06 08:00:00	2023-05-13 19:00:00	35	70.00	14	0110011	1000111
17	2023-04-09 09:00:00	2023-04-18 20:00:00	20	600.00	15	0110100	1001000
18	2023-05-23 10:00:00	2023-05-28 21:00:00	30	2000.00	18	0110101	1001001
19	2023-07-15 11:00:00	2023-07-18 22:00:00	15	150.00	19	0110110	1001010
20	2023-07-21 12:00:00	2023-07-28 23:00:00	20	600.00	20	0110111	1001011
27	2023-05-05 09:00:00	2023-10-05 18:00:00	10	500.00	1	0100100	0111000
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

SCHEMAS Filter objects

- language
- log\_event
- log\_reservation
- log\_travel\_to
- log\_trip
- manages
- offers
- phone
- reservation
- reservation\_offers
- travel\_to
- trip
- worker

Administration Schemas

Information Schema: travel\_agency

Query 1 SQL File 3

```
1 • call check_trip(1,'23-01-05','2023-10-05');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

kvdikos_upokatasthmatos	kvdikos_taksidou	kostos_taksidou	megistes_theseis	hm_anaxirwshs	hm_epistrofhs	kenesthesiai
1	1	500.00	10	2023-01-05 09:00:00	2023-01-05 18:00:00	9

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Query 1 SQL File 3\*

Schemas: language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, offers, phone, reservation, reservation\_offers, travel\_to, trip, worker

Administration Schemas

Information: Schema: travel\_agency

Result Grid | Filter Rows: Export: Wrap Cell Content:

	kwdikos_upokatasthatmos	kwdikos_taksidiou	onomaepwnumo_odehgou
1	1	1	lampri lamprou

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Query 1 SQL File 3\*

Schemas: language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, offers, phone, reservation, reservation\_offers, travel\_to, trip, worker

Administration Schemas

Information: Schema: travel\_agency

Result Grid | Filter Rows: Export: Wrap Cell Content:

	kwdikos_upokatasthatmos	kwdikos_taksidiou	onomaepwnumo_ksenagou
1	1	1	olga xasekidou

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

SCHEMAS

language log\_event log\_reservation log\_travel\_to log\_trip manages offers phone reservation reservation\_offers travel\_to trip worker

Administration Schemas

Information

Schema: travel\_agency

Query 1 SQL File 3\*

Result Grid Filter Rows: Export: Wrap Cell Content:

	kwdikos_upokatasthamatos	kwdikos_taksidou	kostos_taksidou	megistes_theseis	hm_anaxwirhs	hm_epistrofhs	kenestheiseis
1	27	500.00	10	2023-05-05 09:00:00	2023-10-05 18:00:00	9	

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

SCHEMAS

language log\_event log\_reservation log\_travel\_to log\_trip manages offers phone reservation reservation\_offers travel\_to trip worker

Administration Schemas

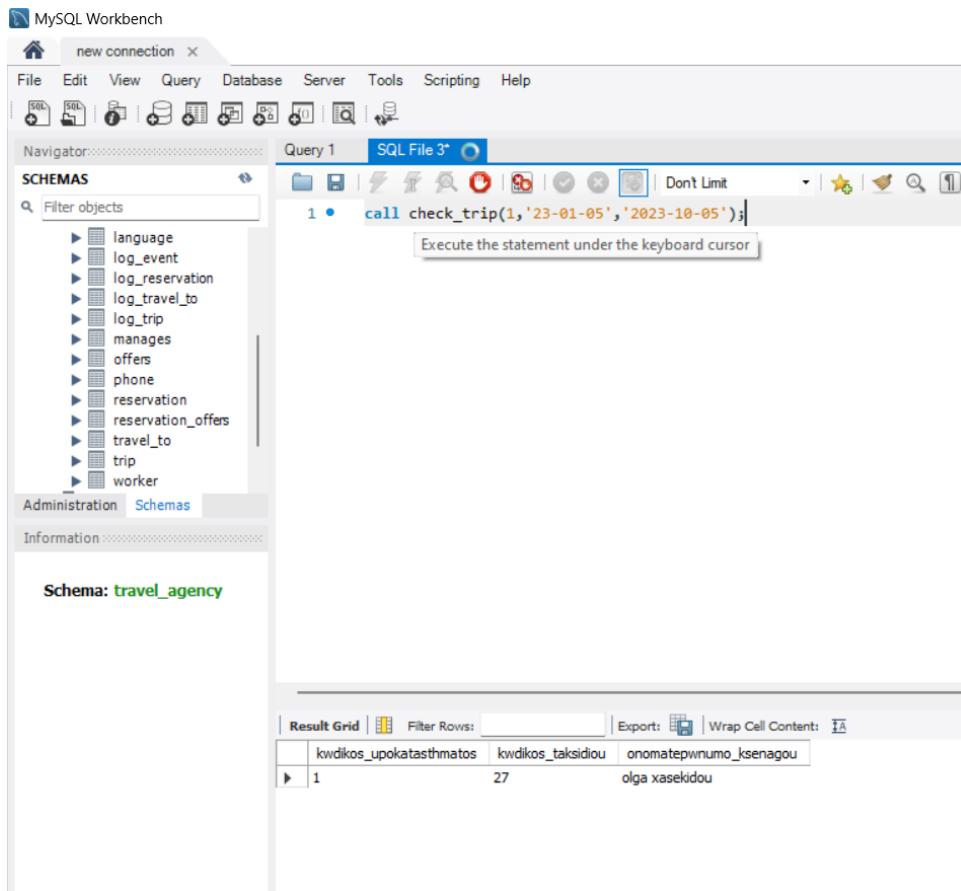
Information

Schema: travel\_agency

Query 1 SQL File 3\*

Result Grid Filter Rows: Export: Wrap Cell Content:

	kwdikos_upokatasthamatos	kwdikos_taksidou	onomatopwnumo_odhgou
1	27	lamprou	lamprini



### 3.1.3.4

`CREATE INDEX id_nlname ON reservation_offers(name,lname);`

//Επιλέγω να δημιουργήσω ευρετήριο ,βάση του ονόματος και του επιθέτου, διότι πολλές φορές θα χρειαστεί να κάνω μια αναζήτηση στον πίνακα reservation\_offers βάση του ονόματος και όχι του res\_offer\_id.

//Παρατηρώ την διαφορά στους χρόνους εκτέλεσης ενός τυχαίου ερωτήματος .Αυτό οφείλεται στο ότι τα ευρητήρια δημιουργούν μικρότερους πίνακες από τις καθορισμένες στηλες name,lname του πίνακα reservation\_offers. Δηλαδή μεταφέρουμε ένα μη ταξινομημένο πίνακα σε μια σειρά που θα μεγιστοποιήσει την αποτελεσματικότητα του ερωτήματος κατά την αναζήτηση.

//σε κάποια ερωτήματα στη συνέχεια ,θα εμφανίζονται τα ίδια ονόματα γιατί έχω περάσει τα 1000\*60 φορές

//0.031 sec

```

MySQL Workbench
File Edit View Query Database Server Tools Help
Navigator: Schemas
Table: reservation_offers
Columns:
offer_id int AI PK
name varchar(50)
lname varchar(50)
offer_id int PK
deposit int
Result Grid | Filter Rows: Export: Wrap Cell Content: 
deposit
113
111
161
62
148
reservation_offers10.x
Output
Action Output
# Time Action Message Duration / Fetch
1 15:16:15 select deposit from reservation_offers where name='Aurora' and lname='Hensley' 61 row(s) returned 0.031 sec / 0 sec

```

//0.00 sec

```

MySQL Workbench
File Edit View Query Database Server Tools Help
Navigator: Schemas
Table: reservation_offers
Columns:
offer_id int AI PK
name varchar(50)
lname varchar(50)
offer_id int PK
deposit int
Result Grid | Filter Rows: Export: Wrap Cell Content: 
deposit
113
111
161
62
148
reservation_offers11.x
Output
Action Output
# Time Action Message Duration / Fetch
1 15:16:15 select deposit from reservation_offers where name='Aurora' and lname='Hensley' 61 row(s) returned 0.031 sec / 0.000 sec
2 15:16:51 CREATE INDEX id_lname ON reservation_offers(name,lname) 0 rows affected Records: 0 Duplicates: 0 Warnings: 0 0.030 sec
3 15:16:55 select deposit from reservation_offers where name='Aurora' and lname='Hensley' 61 row(s) returned 0.000 sec / 0.000 sec

```

a.

DELIMITER \$

```

CREATE PROCEDURE check_deposit( IN num1 INT, IN num2 INT )
BEGIN
DECLARE name_offer VARCHAR(20);
DECLARE lname_offer VARCHAR(20);
DECLARE deposit_offer INT;
DECLARE flag INT ;

```

```

DECLARE cr CURSOR FOR
SELECT name,lname,deposit
FROM reservation_offers
WHERE deposit BETWEEN num1 AND num2;

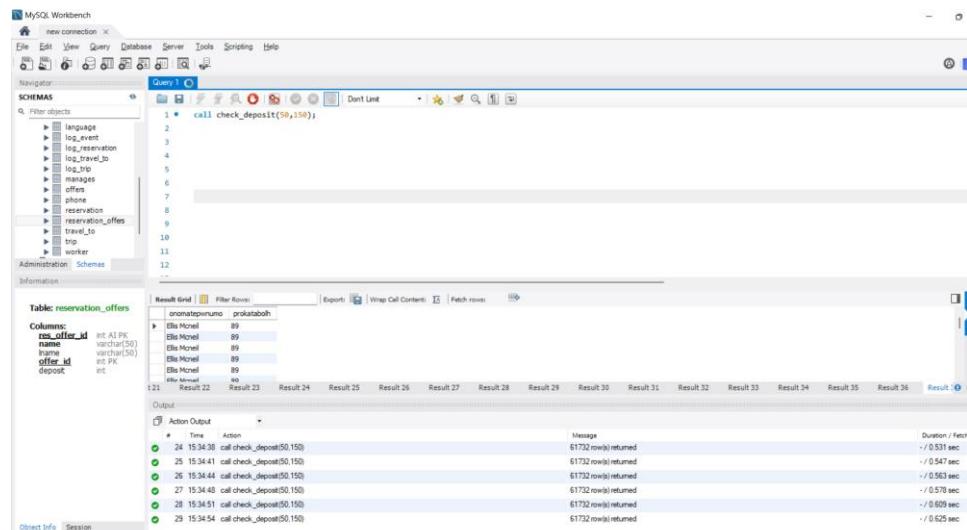
```

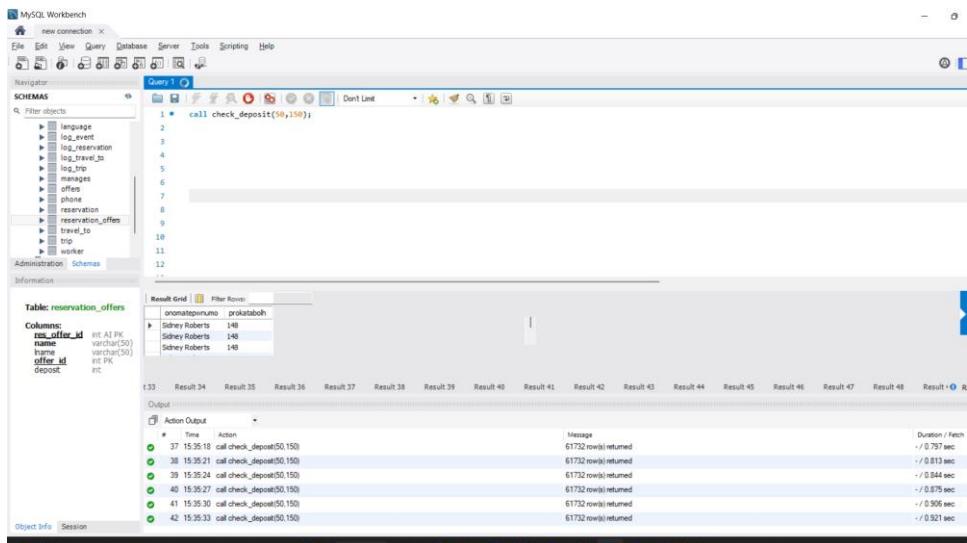
```

DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag=1;

OPEN cr;
SET flag=0;
FETCH cr INTO name_offer,lname_offer,deposit_offer;
WHILE(flag=0) DO
  FETCH cr INTO name_offer,lname_offer,deposit_offer;
  SELECT CONCAT(name_offer,' ',lname_offer) AS onomatepwnumo, deposit_offer AS prokatabolh
  FROM reservation_offers
  WHERE deposit_offer BETWEEN num1 AND num2;
  DO SLEEP(3);
END WHILE;
CLOSE cr;
END$
```

DELIMITER;





β.

```

CREATE PROCEDURE lname_offer( IN lname_offer VARCHAR(20))

BEGIN

DECLARE myname VARCHAR(20);

DECLARE mylname VARCHAR(20);

DECLARE myofferid INT ;

DECLARE flag INT;

DECLARE cr CURSOR FOR

SELECT name,lname,offer_id FROM reservation_offers WHERE lname=lname_offer;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag=1;

OPEN cr;

SET flag=0;

WHILE(flag=0) DO

FETCH cr INTO myname,mylname,myofferid;

SELECT CONCAT(myname,' ',mylname) AS onomatepwnumo,myofferid AS kwdikos_prosforas

FROM reservation_offers

```

```
WHERE lname=mylname;
```

```
DO SLEEP(1);
```

```
END WHILE;
```

```
CLOSE cr;
```

```
END$
```

```
DELIMITER;
```

```
//Εστω ότι θέλω να βρω τους πελάτες/πελάτη με το επίθετο 'Whitehead' και την προσφορά  
ταξιδιού που έχουν δηλώσει.
```

The screenshot shows the MySQL Workbench interface with two tabs: 'Query 1' and 'Query 2'. Both tabs contain the same SQL code:

```
1 |  
2 • call lname_offer('whitehead')  
3 |
```

The 'Result Grid' pane displays the output of the stored procedure. It shows a single row with the following data:

onoma	tegnwno	leidiko_proforas
Natalia Whitehead	1	

The 'Object Info' and 'Session' tabs are also visible at the bottom of the interface.

The screenshot shows the MySQL Workbench interface with two tabs: 'Query 1' and 'Query 2'. Both tabs contain the same SQL code:

```
1 |  
2 • call lname_offer('whitehead')  
3 |
```

The 'Result Grid' pane displays the output of the stored procedure. It shows a single row with the following data:

onoma	tegnwno	leidiko_proforas
Natalia Whitehead	2	

The 'Object Info' and 'Session' tabs are also visible at the bottom of the interface.

---

### *Κεφάλαιο 3*

#### **TRIGGERS :**

##### **3.1.4.1 // δεν έχω κάνει την σύνδεση ώστε να δείχνει ποιος εκτέλεσε μία ενέργεια**

###### **α. table trip**

1. DELIMITER \$ // insert

```
CREATE TRIGGER trip_insert_log_trigger
```

```
AFTER INSERT ON trip FOR EACH ROW
```

```
BEGIN
```

```
    INSERT INTO log_trip (
```

```
        trip_id,
```

```
        old_row,
```

```
        new_row,
```

```
        type_action,
```

```
        time_action
```

```
)
```

```
VALUES(
```

```
    NEW.tr_id,
```

```
    null,
```

```
    JSON_OBJECT(
```

```
        "tr_departure", NEW.tr_departure,
```

```
        "tr_return", NEW.tr_return,
```

```
        "tr_maxseats", NEW.tr_maxseats,
```

```
        "tr_cost", NEW.tr_cost,
```

```
        "tr_br_code", NEW.tr_br_code,
```

```
        "tr_gui_AT", NEW.tr_gui_AT,
```

```

        "tr_drv_AT",NEW.tr_drv_AT),
        'INSERT',
CURRENT_TIMESTAMP
);
END $
```

DELIMITER ;

//Κάνω μία εισαγωγή για να δω πως τρέχει ο πίνακας log\_trip

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the database schema with various tables like language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, offers, phone, reservation, reservation\_offers, travel\_to, trip, and worker. The 'reservation\_offers' table is selected. In the center, the Query Grid contains the following SQL code:

```

1
2 • INSERT INTO trip(tr_departure,tr_return,tr_maxseats,tr_cost,tr_br_code,tr_gui_AT,tr_drv_AT)
3 VALUES('2023-01-08 20:00:00','2023-04-04 12:00:00',30,800,2,'0100111','0111000')
4 • select * from log_trip;
```

The Result Grid shows one row of data from the log\_trip table:

trip_id	old_row	new_row	type_action	time_action
28		{"tr_cost": 800.0, "tr_drv_AT": "0111000", "tr_gui_AT": "0100111"}	INSERT	2023-02-02 16:27:29

## 2. CREATE TRIGGER trip\_update\_log\_trigger //update

AFTER UPDATE ON trip FOR EACH ROW

BEGIN

    INSERT INTO log\_trip (

        trip\_id,

        old\_row,

        new\_row,

        type\_action,

        time\_action

)

VALUES(

    NEW.trip\_id,

```

JSON_OBJECT(
    "tr_departure", OLD.tr_departure,
    "tr_return", OLD.tr_return,
    "tr_maxseats", OLD.tr_maxseats,
    "tr_cost", OLD.tr_cost,
    "tr_br_code",OLD.tr_br_code,
    "tr_gui_AT", OLD.tr_gui_AT,
    "tr_drv_AT",OLD.tr_drv_AT
),
JSON_OBJECT(
    "tr_departure", NEW.tr_departure,
    "tr_return", NEW.tr_return,
    "tr_maxseats", NEW.tr_maxseats,
    "tr_cost", NEW.tr_cost,
    "tr_br_code",NEW.tr_br_code,
    "tr_gui_AT", NEW.tr_gui_AT,
    "tr_drv_AT",NEW.tr_drv_AT
),
'UPDATE',
CURRENT_TIMESTAMP
);
END $$
DELIMITER ;

```

**//Παρατηρώ την τελευταία εισαγωγή με tr\_id=30**

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with icons for new connection, file operations, and search. The left sidebar has sections for Navigator, SCHEMAS, and Administration, with 'Schemas' currently selected. A tree view under 'schemas' lists tables: branch, destination, driver, event, guide, it, language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, and offers. The main area has a 'Query 1' tab titled 'SQL File 4\*' showing the query 'select \* from trips;'. The results grid below shows data for 30 rows of the 'trips' table, with columns: tr\_id, tr\_departure, tr\_return, tr\_maxseats, tr\_cost, tr\_br\_code, tr\_gu\_AT, and tr\_drv\_AT. The data includes various travel dates and costs.

tr_id	tr_departure	tr_return	tr_maxseats	tr_cost	tr_br_code	tr_gu_AT	tr_drv_AT
19	2023-07-15 11:00:00	2023-07-18 22:00:00	15	150.00	19	0110110	1001010
20	2023-07-21 12:00:00	2023-07-28 23:00:00	20	600.00	20	0110111	1001011
27	2023-05-05 09:00:00	2023-10-05 18:00:00	10	500.00	1	0100100	0111000
30	2023-01-08 20:00:00	2023-04-04 12:00:00	30	10000.00	2	0100111	0111000

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for database management. The left sidebar has sections for Navigator, SCHEMAS, and Administration. The SCHEMAS section lists tables: branch, destination, driver, event, guide, it, language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, and offers. The Administration section shows 'Schemas' selected. The main area contains a 'Query 1' tab with the SQL code:

```

1 • update trip set tr_cost=60 where tr_id=30;
2 • select * from log_trip;
3
4
5
6
7
8
9
10
11
12
--
```

Below the code is a 'Result Grid' table with columns: trip\_id, old\_row, new\_row, type\_action, and time\_action. The data shows an UPDATE operation for trip\_id 30:

trip_id	old_row	new_row	type_action	time_action
30	{"tr_cost": 10000.0, "tr_drv_AT": "01110000", "tr_g...}	{"tr_cost": 60.0, "tr_drv_AT": "01110000", "tr_g...}	UPDATE	2023-02-02 16:45:49

At the bottom, a status bar indicates 'log\_trip 212 x'.

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar, titled 'Navigator', contains sections for SCHEMAS and TABLES. Under SCHEMAS, there are 14 entries: branch, destination, driver, event, guide, it, language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, and offers. Under TABLES, there is one entry: reservation\_offers. The central area has tabs for 'Query 1' and 'SQL File 4'. The 'Query 1' tab displays the following SQL code:

```

1 * select * from trip;
2
3
4
5
6
7
8
9
10
11
12

```

The 'Result Grid' tab shows the data from the 'reservation\_offers' table. The columns are: tr\_id, tr\_departure, tr\_return, tr\_maxseats, tr\_cost, tr\_br\_code, tr\_gu\_AT, and tr\_drv\_AT. The data is as follows:

tr_id	tr_departure	tr_return	tr_maxseats	tr_cost	tr_br_code	tr_gu_AT	tr_drv_AT
19	2023-07-15 11:00:00	2023-07-18 22:00:00	15	150.00	19	0110110	1001010
20	2023-07-21 12:00:00	2023-07-28 23:00:00	20	600.00	20	0110111	1001011
27	2023-05-09 09:00:00	2023-10-15 18:00:00	10	500.00	1	0100100	0114000
*	2023-01-08 20:00:00	2023-04-04 12:00:00	30	60.00	2	0100111	0111000
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

A red box highlights the last two rows of the result grid. Below the grid, a status bar shows 'trip 235'.

### 3. DELIMITER \$ //delete

CREATE TRIGGER trip delete log trigger

AFTER DELETE ON trip FOR EACH ROW

BEGIN

```
INSERT INTO log_trip(
    trip_id,
    old_row,
    new_row,
    type_action,
    time_action
)
VALUES(
    OLD.tr_id,
    JSON_OBJECT(
        "tr_departure", OLD.tr_departure,
        "tr_return", OLD.tr_return,
        "tr_maxseats", OLD.tr_maxseats,
        "tr_cost", OLD.tr_cost,
        "tr_br_code", OLD.tr_br_code,
        "tr_gui_AT", OLD.tr_gui_AT,
        "tr_drv_AT", OLD.tr_drv_AT
    ),
    null,
    'DELETE',
    CURRENT_TIMESTAMP
);
END $
```

DELIMITER ;

The screenshot shows the MySQL Workbench interface. In the top-left, there's a 'Navigator' pane listing various database objects like branch, destination, driver, event, guide, it, language, log\_event, log\_reservation, log\_travel\_to, log\_trip, manages, and offers. Below the navigator is an 'Information' pane showing the 'reservation\_offers' table with columns: res\_offer\_id (int AI PK), name (varchar(50)), lname (varchar(50)), offer\_id (int PK), and name. The main area contains a 'Query 1' tab with the following SQL code:

```

1 • delete from trip where tr_id=30;
2 • select * from log_trip;
3
4
5
6
7
8
9
10
11
12

```

Below the code is a 'Result Grid' showing the output of the 'select \* from log\_trip;' statement. The grid has columns: trip\_id, old\_row, new\_row, type\_action, and time\_action. It displays two rows of data:

trip_id	old_row	new_row	type_action	time_action
30	{"tr_cost": 10000.0, "tr_drv_AT": "0111000", "tr_g_AT": "0111000", "tr_g_id": null, "tr_id": null}	{"tr_cost": 60.0, "tr_drv_AT": "0111000", "tr_g_AT": "0111000", "tr_g_id": null, "tr_id": null}	UPDATE	2023-02-02 16:45:49
30	null	null	DELETE	2023-02-02 16:49:09

\*\*\*Αντίστοιχα και στους επόμενους log πίνακες reservation,travel\_to,event

### β. table reservation

#### 1. DELIMITER \$

```
CREATE TRIGGER rsv_insert_log_trigger //insert
```

```
AFTER INSERT ON reservation FOR EACH ROW
```

```
BEGIN
```

```
    INSERT INTO log_reservation (
```

```
        res_trip_id,
```

```
        old_row,
```

```
        new_row,
```

```
        type_action,
```

```
        time_action
```

```
)
```

```
    VALUES(
```

```
        NEW.res_tr_id,
```

```
        null,
```

```
        JSON_OBJECT(
```

```
            "res_tr_id", NEW.res_tr_id,
```

```
            "res_seatnum", NEW.res_seatnum,
```

```
            "res_name", NEW.res_name,
```

```
"res_lname", NEW.res_lname,  
"res_isadult",NEW.res_isadult  
,  
'INSERT',  
CURRENT_TIMESTAMP  
);  
END $  
DELIMITER ;
```

## 2. DELIMITER \$ //update

```
CREATE TRIGGER rsv_update_log_trigger  
AFTER UPDATE ON reservation FOR EACH ROW  
BEGIN  
    INSERT INTO log_reservation (  
        res_trip_id,  
        old_row,  
        new_row,  
        type_action,  
        time_action  
    )  
    VALUES(  
        NEW.res_tr_id,  
        JSON_OBJECT(  
            "res_tr_id", OLD.res_tr_id,  
            "res_seatnum", OLD.res_seatnum,  
            "res_name", OLD.res_name,  
            "res_lname", OLD.res_lname,  
            "res_isadult",OLD.res_isadult  
,  
        ),
```

```

JSON_OBJECT(
    "res_tr_id", NEW.res_tr_id,
    "res_seatnum", NEW.res_seatnum,
    "res_name", NEW.res_name,
    "res_lname", NEW.res_lname,
    "res_isadult", NEW.res_isadult
),
'UPDATE',
CURRENT_TIMESTAMP
);

```

END \$

DELIMITER ;

### 3. DELIMITER \$ //delete

```

CREATE TRIGGER rsv_delete_log_trigger
AFTER DELETE ON reservation FOR EACH ROW
BEGIN
    INSERT INTO log_reservation(
        res_trip_id,
        old_row,
        new_row,
        type_action,
        time_action
    )
    VALUES(
        OLD.res_tr_id,
        JSON_OBJECT(
            "res_tr_id", OLD.res_tr_id,
            "res_seatnum", OLD.res_seatnum,

```

```
"res_name", OLD.res_name,  
"res_lname", OLD.res_lname,  
"res_isadult",OLD.res_isadult  
,  
null,  
'DELETE',  
CURRENT_TIMESTAMP  
);  
END $  
DELIMITER ;
```

---

## y. table event

```
1. DELIMITER $ //insert  
CREATE TRIGGER event_insert_log_trigger  
AFTER INSERT ON event FOR EACH ROW  
BEGIN  
INSERT INTO log_event (  
event_trip_id,  
old_row,  
new_row,  
type_action,  
time_action  
)  
VALUES(  
NEW.ev_tr_id,  
null,  
JSON_OBJECT(
```

```
"ev_start", NEW.ev_start,  
"ev_end", NEW.ev_end,  
"ev_descr", NEW.ev_descr  
  
,  
'INSERT',  
CURRENT_TIMESTAMP  
);  
END $  
DELIMITER ;
```

## 2. DELIMITER \$ //update

```
CREATE TRIGGER event_update_log_trigger  
AFTER UPDATE ON event FOR EACH ROW  
BEGIN  
    INSERT INTO log_event (  
        event_trip_id,  
        old_row,  
        new_row,  
        type_action,  
        time_action  
    )  
    VALUES(  
        NEW.ev_tr_id,  
        JSON_OBJECT(  
            "ev_start", OLD.ev_start,  
            "ev_end", OLD.ev_end,  
            "ev_descr", OLD.ev_descr
```

```
        ),
        JSON_OBJECT(
    "ev_start", NEW.ev_start,
    "ev_end", NEW.ev_end,
    "ev_descr", NEW.ev_descr
),
'UPDATE',
CURRENT_TIMESTAMP
);
END $
```

DELIMITER ;

### 3. DELIMITER \$ //delete

```
CREATE TRIGGER event_delete_log_trigger
AFTER DELETE ON event FOR EACH ROW
BEGIN
    INSERT INTO log_event(
        event_trip_id,
        old_row,
        new_row,
        type_action,
        time_action
    )
    VALUES(
        OLD.ev_tr_id,
        JSON_OBJECT(
    "ev_start", OLD.ev_start,
```

```
"ev_end", OLD.ev_end,  
"ev_descr", OLD.ev_descr  
,  
null,  
'DELETE',  
CURRENT_TIMESTAMP  
);  
END $  
DELIMITER ;
```

---

## δ. table travel\_to

1. DELIMITER \$\$ //insert

```
CREATE TRIGGER trto_insert_log_trigger  
AFTER INSERT ON travel_to FOR EACH ROW  
BEGIN  
INSERT INTO log_travel_to (  
    trto_trip_id,  
    old_row,  
    new_row,  
    type_action,  
    time_action  
)  
VALUES(  
    NEW.to_tr_id,  
    null,  
    JSON_OBJECT(  
        "to_dst_id", NEW.to_dst_id,  
        "to_arrival", NEW.to_arrival,
```

```
"to_departure", NEW.to_departure  
  
,  
'INSERT',  
CURRENT_TIMESTAMP  
);  
END $  
DELIMITER ;
```

## 2. DELIMITER \$\$ //update

```
CREATE TRIGGER trto_update_log_trigger  
AFTER UPDATE ON travel_to FOR EACH ROW  
BEGIN  
    INSERT INTO log_travel_to (  
        trto_trip_id,  
        old_row,  
        new_row,  
        type_action,  
        time_action  
    )  
    VALUES(  
        NEW.to_tr_id,  
        JSON_OBJECT(  
            "to_dst_id", OLD.to_dst_id,  
            "to_arrival", OLD.to_arrival,  
            "to_departure", OLD.to_departure  
        ),  
        JSON_OBJECT(  
    );
```

```

"to_dst_id", NEW.to_dst_id,
"to_arrival", NEW.to_arrival,
"to_departure", NEW.to_departure
),
'UPDATE',
CURRENT_TIMESTAMP
);

END $

DELIMITER ;

```

### 3. DELIMITER \$ //delete

```

CREATE TRIGGER trto_delete_log_trigger
AFTER DELETE ON travel_to FOR EACH ROW
BEGIN
INSERT INTO log_travel_to(
trto_trip_id,
old_row,
new_row,
type_action,
time_action
)
VALUES(
OLD.to_tr_id,
JSON_OBJECT(
"to_dst_id", OLD.to_dst_id,
"to_arrival", OLD.to_arrival,
"to_departure", OLD.to_departure
),
null,

```

```

'DELETE',
CURRENT_TIMESTAMP
);

END $

DELIMITER ;

```

### 3.1.4.3

**a.**

```

DELIMITER $

CREATE TRIGGER salary
BEFORE UPDATE ON worker
FOR EACH ROW
BEGIN

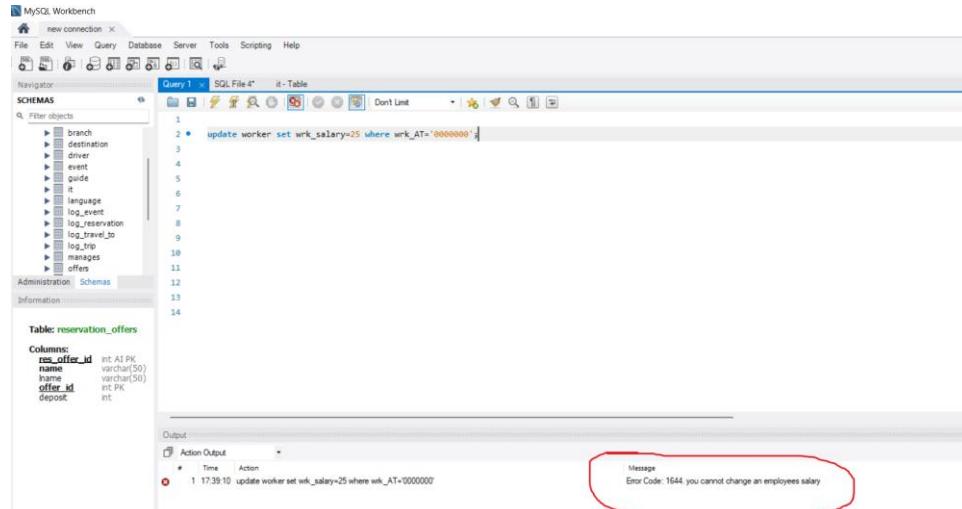
```

```

IF NEW.wrk_salary NOT LIKE OLD.wrk_salary THEN
SIGNAL SQLSTATE VALUE '45000'
SET MESSAGE_TEXT = 'YOU CANT CHANGE AN EMPLOYEES SALARY';
END IF;
END $

```

DELIMITER;



**β.**

```
DELIMITER $  
  
CREATE TRIGGER prevent_trip_changes  
BEFORE UPDATE ON trip  
FOR EACH ROW  
BEGIN  
DECLARE reservation_count INT;  
  
SELECT COUNT(*) INTO reservation_count  
FROM reservation  
WHERE res_tr_id = OLD.tr_id;  
  
IF reservation_count > 0 THEN  
IF NEW.tr_departure != OLD.tr_departure OR  
NEW.tr_return != OLD.tr_return OR  
NEW.tr_cost != OLD.tr_cost THEN  
SIGNAL SQLSTATE "45000" SET MESSAGE_TEXT = "Exei ginei kratisi gia afto to trip";  
END IF;  
END IF;  
END$  
  
DELIMITER ;
```

**//πχ αν θελω να αλλαξω την ημερομηνια του ταξιδιου με tr\_id=1**

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the schema 'travel\_agency', there is a table named 'trip'. A query in the 'Query 1' editor window is run:

```
1 • select * from trip where tr_id=1;
```

The result grid displays one row of data:

tr_id	tr_departure	tr_return	tr_maxseats	tr_cost	tr_br_code	tr_gui_AT	tr_drv_AT
1	2023-01-05 09:00:00	2023-01-05 18:00:00	10	500.00	1	0100100	0111000

//μου εμφανίζει κατάλληλο μήνυμα ότι δεν είναι δυνατή αυτή η ενέργεια

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the schema 'travel\_agency', there is a table named 'trip'. A query in the 'Query 1' editor window is run:

```
1 • update trip set tr_departure='2023-01-06 13:00:00';
```

The 'Output' pane shows the execution log:

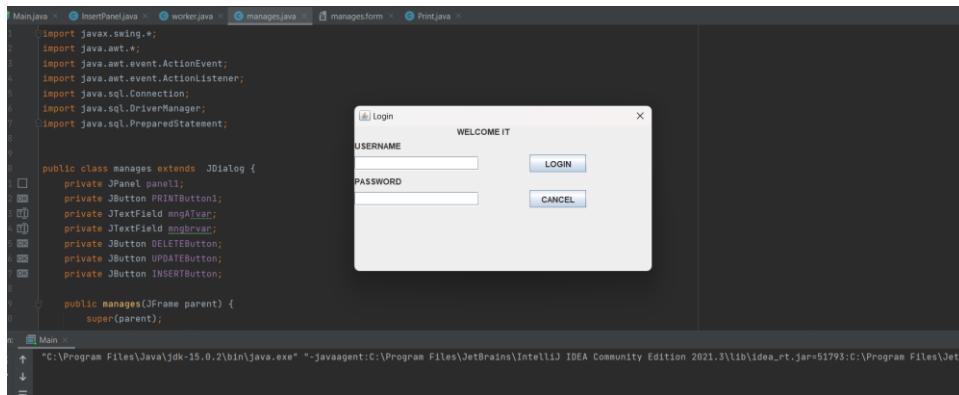
Action	Time	Action	Message
1	22:17:23	CREATE TRIGGER prevent_trip_changes BEFORE UPDATE ON trip FOR EACH ROW BEGIN DECLARE res...	Error Code: 1064. You have an error in your SQL syntax; check the manual that came...
2	22:17:38	CREATE TRIGGER prevent_trip_changes BEFORE UPDATE ON trip FOR EACH ROW BEGIN DECLARE res...	0 row(s) affected
3	22:19:05	select * from trip	21 row(s) returned
4	22:19:26	select * from trip where tr_id=1	1 row(s) returned
5	22:22:25	update trip set tr_departure='2023-01-06 13:00:00'	Error Code: 1044. Erre ginei kállis ga alto to trip

## Β Μέρος:

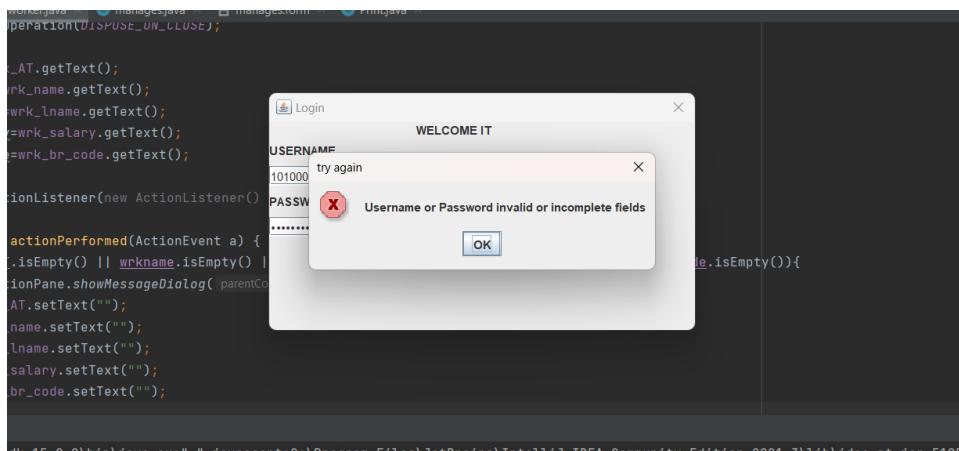
\*η εκτύπωση των πινάκων δεν μου τρέχει οπότε θα σας δείξω τα αποτελέσματα στο mysql workbench

\*τα delete/update θα λειτουργούν ακόμα και αν ο it δεν βάλει όλα τα στοιχεία

//Στο login panel θα συνδέεται ο κάθε it με το AT.



//σε περίπτωση που το όνομα ή ο κωδικός δεν είναι σωστά ή δεν έχουν συμπληρωθεί όλα τα στοιχεία



//Θα διαλέγω ένα πίνακα που θέλω να κάνω μία ενέργεια πχ τον worker

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project:** TRAVEL\_AGENCY\_TEST
- File:** managers.java
- Code:** The code defines a class named `manages` that extends `JDialog`. It contains fields for a panel, a print button, and two text fields (`mngAtVar` and `mngBrVar`). It also includes methods for `DELETEButton`, `UPDATEButton`, `INSERTButton`, and a constructor.
- Run:** The run configuration is set to "Main".
- Output:** The terminal shows the command being run: "C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea\_rt.jar=55144:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8 Main". It also shows a warning about SSL and successful authentication.
- Modals:** A modal titled "ACTION" is displayed, showing a table of 18 rows with columns for "ACTION" and "RESULT". All entries show "OK".

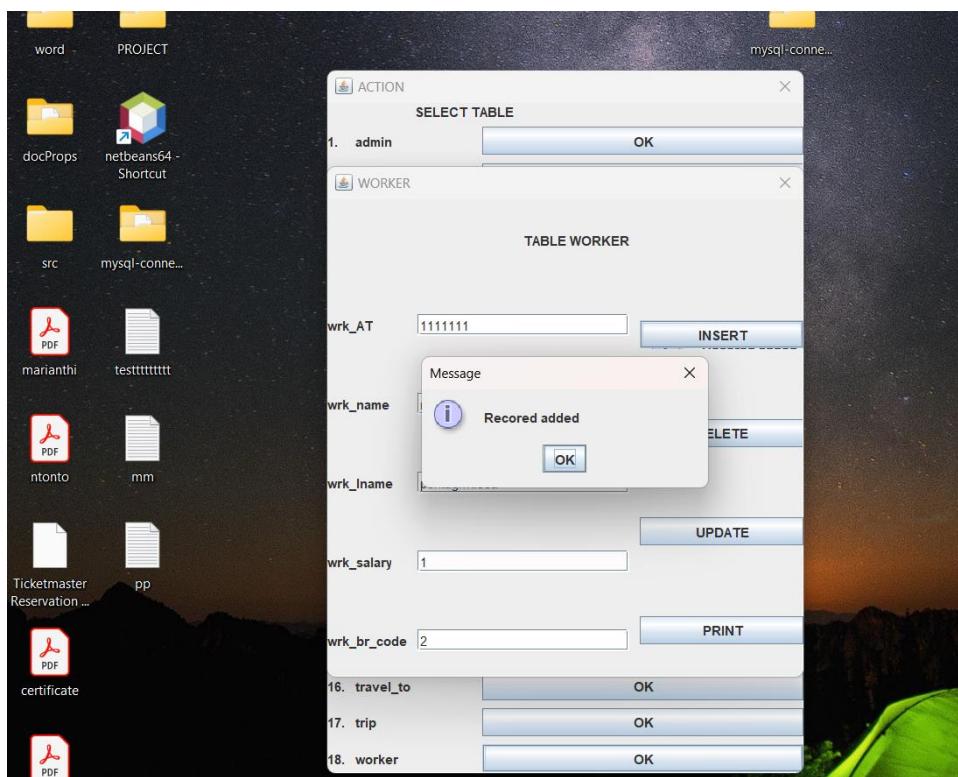
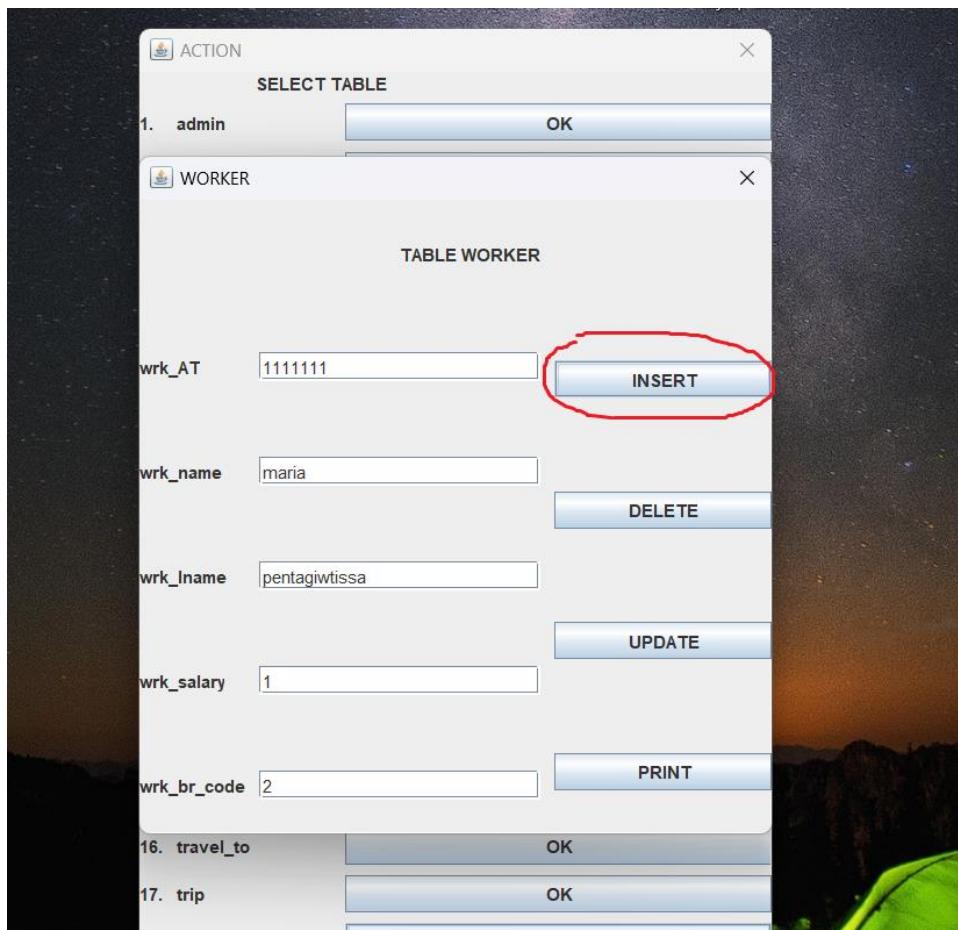
ACTION	RESULT
1. admin	OK
2. branch	OK
3. destination	OK
4. driver	OK
5. event	OK
6. guide	OK
7. it	OK
8. language	OK
9. log_reservation	OK
10. log_trip	OK
11. manages	OK
12. offers	OK
13. phone	OK
14. reservation	OK
15. reservation_offers	OK
16. travel_to	OK
17. trip	OK
18. worker	OK

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for database management. On the left, the Navigator pane displays the 'travel\_agency' schema, which contains tables, views, stored procedures, and functions. The main area is titled 'Query 1' and contains the SQL command: 'select \* from worker;'. The results are displayed in a 'Result Grid' table:

wrk_AT	wrk_name	wrk_lname	wrk_salary	wrk_br_code
1000100	xrhostos	kinikhs	10	12
1000101	augoustinos	thrampas	10	13
1000110	eleonora	alantou	6	14
1000111	lemonia	ksudatou	2	16
1001000	elenh	lenio	4	17
1001001	lampirni	stafuli	4	18
1001010	iwshf	basilladhs	2	19
1001011	newton	thodis	2	20
1001100	marina	xesou	6	16
1001101	giwrgos	nikas	6	17
1001110	tasos	papanastasiou	6	18
1001111	aleksia	xristopoulou	6	19
1010000	thanasis	mpifteklis	6	20
1010001	marianthi	thodi	10	20
*	HULL	HULL	HULL	HULL

The bottom navigation bar includes tabs for 'worker 1' and 'Output'.

//insert



The screenshot shows the MySQL Workbench interface. In the top-left pane, there's a tree view with nodes like 'hemas' and 'el\_agency'. Below it, a 'Procedures' section is visible. The main area contains a 'Query 1' tab with the SQL command 'select \* from worker;'. The results are displayed in a 'Result Grid' table:

	wrk_AT	wrk_name	wrk_lname	wrk_salary	wrk_br_code
1001010	iwshf	basiliadhs	2	19	
1001011	newton	thodis	2	20	
1001100	marina	xesou	6	16	
1001101	giwrgos	nikas	6	17	
1001110	tasos	papanastasiou	6	18	
1001111	aleksia	xristopoulou	6	19	
1010000	tharanasis	mpitekis	6	20	
1010001	marianthi	thodi	10	20	
*	1111111	maria	pentagiwtissa	1	2

A red oval highlights the last row (id 1111111). Below the grid, an 'Output' window shows the action history:

#	Time	Action
1	13:51:44	select * from worker

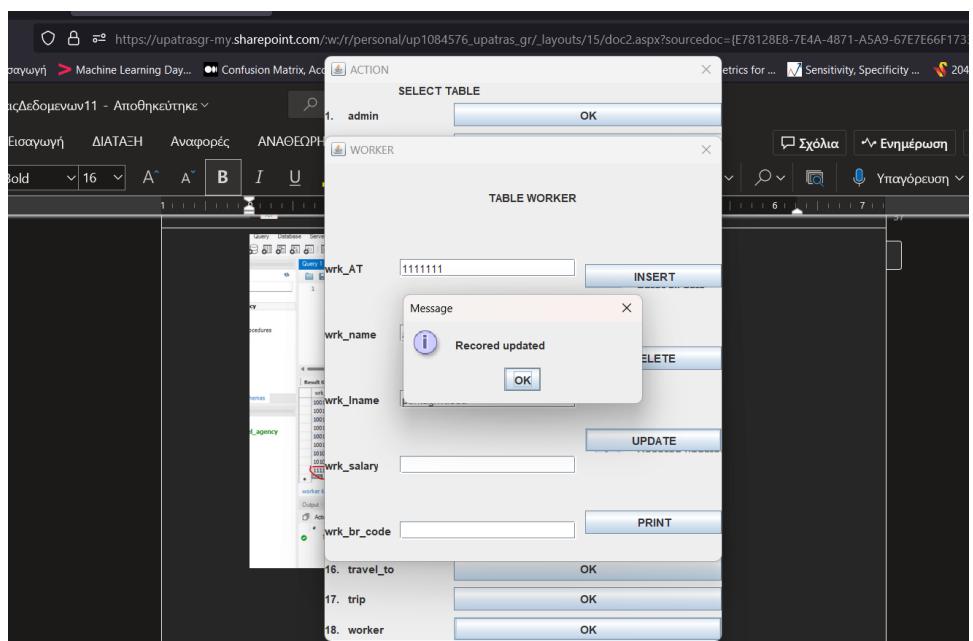
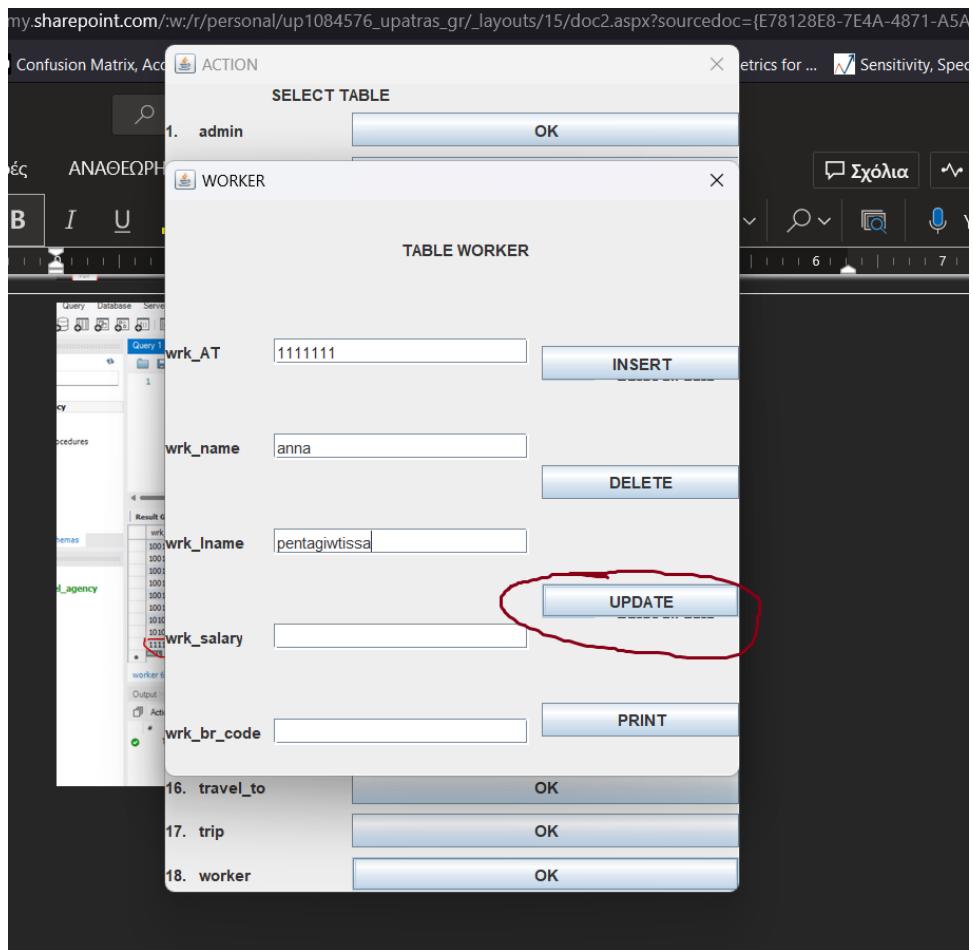
Message: 82 row(s) returned.

//στην περίπτωση του insert πρέπει να τοποθετηθούν όλα τα στοιχεία ενός worker

The screenshot shows a Java application window with code in the background. A modal dialog box titled 'Message' is displayed, containing the text 'Enter all data' with an 'OK' button. The application window has several fields and buttons for interacting with a 'WORKER' table:

- Fields: wrk\_AT, wrk\_name, wrk\_lname, wrk\_salary, wrk\_br\_code.
- Buttons: INSERT, UPDATE, DELETE, PRINT.
- Tables: A 'SELECT TABLE' dropdown with '1. admin' selected, and a 'TABLE WORKER' dropdown.

//τροποποιώ την παραπάνω εγγραφή και από wrk\_name="maria" , το αλλάζω σε wrk\_name="anna".Δεν χρειάζεται η τοποθέτηση όλων των στοιχειών αρκεί το wrk\_AT. // update



Query 1

```
AS
r objects
ys
travel_agency
Tables
Views
Stored Procedures
Functions
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

wrk_AT	wrk_name	wrk_lname	wrk_salary	wrk_br_code
1001010	iwshf	basiliadhs	2	19
1001011	newton	thodis	2	20
1001100	marina	xesou	6	16
1001101	giwrgos	nikas	6	17
1001110	tasos	papanastasiou	6	18
1001111	aleksia	xristopoulou	6	19
1010000	thanasis	mpftekis	6	20
1010001	marianthi	thodi	10	20
1111111	anna	pentagiwtissa	1	1111111

worker 7

Output

//delete

Action

SELECT TABLE

1. admin OK

WORKER X

TABLE WORKER

wrk\_AT 1111111 INSERT

wrk\_name [ ] DELETE (circled)

wrk\_lname [ ] UPDATE

wrk\_salary [ ] PRINT

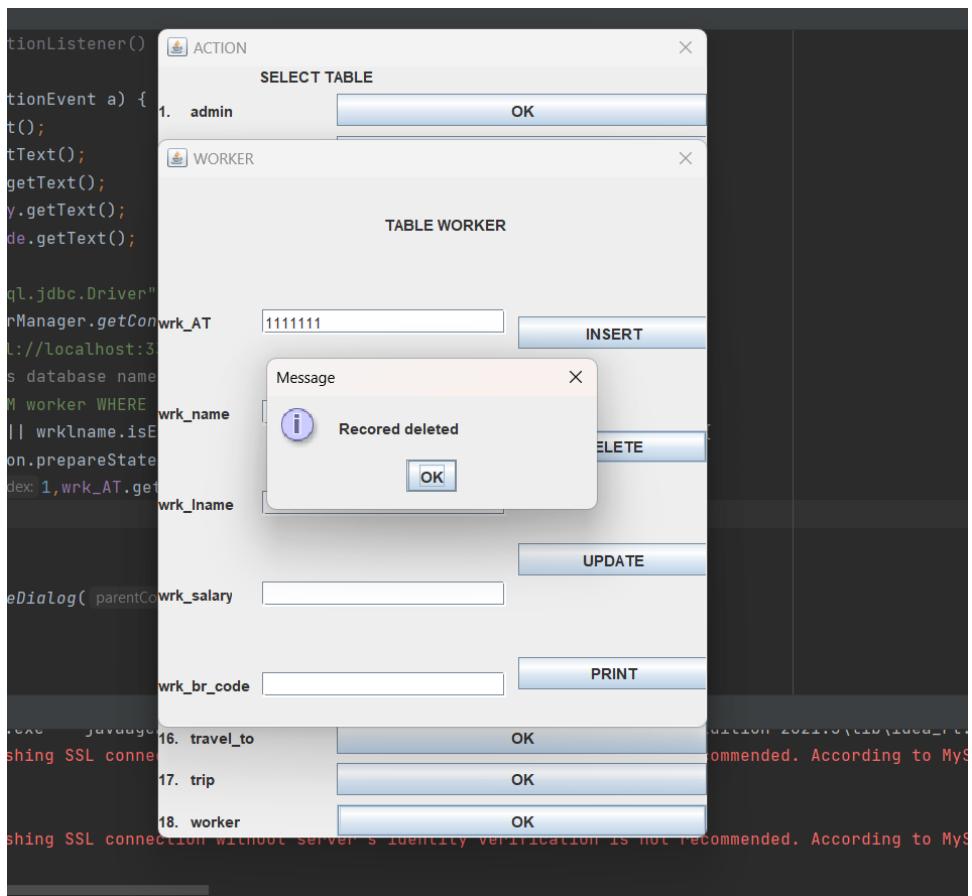
wrk\_br\_code [ ]

16. travel\_to OK

023 WARN: Establishing SSL connection without server certificate verification. Recommended. According to

17. trip OK

18. worker OK



The screenshot shows the MySQL Workbench interface. A query window titled 'Query 1' displays the results of the SQL query 'select \* from workers'. The results are shown in a grid format. One specific row, which corresponds to the deleted record in the Java application, is highlighted with a red box.

wrk_AT	wrk_name	wrk_name	wrk_salary	wrk_br_code
100101	lanyrin	stafis	4	18
100102	lanyrin	gennadios	2	29
100111	newton	rhodis	2	29
100100	marina	veiss	6	16
100101	giorgos	nikas	6	17
100101	giorgos	papadimitris	6	18
100111	delela	vangelopoulos	6	19
100000	theassis	mpithios	6	20
100001	magnolia	rhodis	10	20

Όμοια λοιπόν για όλους τους υπόλοιπους πίνακες.

