3.1.3.4

Στο ερώτημα 3.1.3.4, καλούμαστε να δημιουργήσουμε 2 stored procedures, οι οποίες θα εκτελούν ουσιαστικά μία αναζήτηση στον πίνακα reservation\_offers με βάση κάποιας(-ων) τιμών, είτε εύρους είτε ταιριάσματος (αλφαριθμητικού). Για να επιταχύνουμε την αναζήτηση των 2 αυτών procedures, θα πρέπει να δημιουργήσουμε από ένα κατάλληλο δείκτη για την καθεμία.

Για την πρώτη procedure, αφού έχουμε ερώτημα εύρους, επιλέγουμε να χρησιμοποιήσουμε BTREE index.

Για τη δεύτερη procedure, αφού έχουμε ερωτήματα ισότητας , επιλέγουμε να χρησιμοποιήσουμε HASH index, κάτι που έχει ως αποτέλεσμα να χρειαστεί να αλλάξουμε το ENGINE του πίνακα από το DEFAULT InnoDB σε Memory, καθώς το πρώτο δεν υποστηρίζει HASH type indexes.

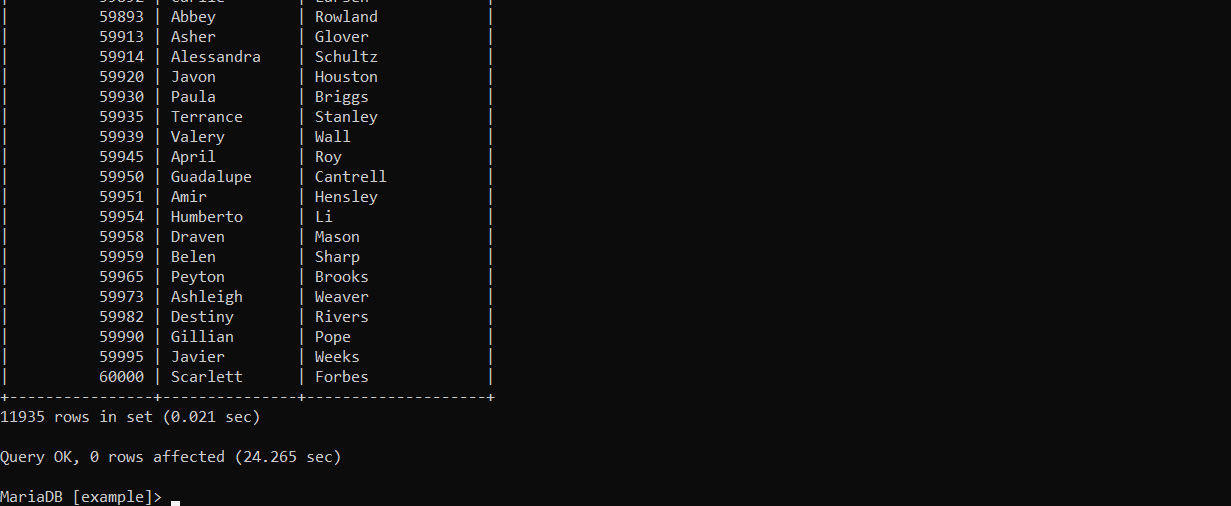
Παρακάτω, παραθέτουμε πρώτα τα αποτελέσματα από τις αναζητήσεις με/χωρίς indexes, και έπειτα τα προβλήματα που έπρεπε να επιλυθούν όταν κάναμε την αλλαγή από InnoDB σε Memory ENGINE στον πίνακα reservation\_offers.

α)

Για την πρώτη procedure, αφού η αναζήτηση που εκτελούμε μέσα σε αυτή, γίνεται με βάση το γνώρισμα advance\_fee, θα δημιουργήσουμε index για αυτό το γνώρισμα στον πίνακα reservation\_offers.

-> CALL get\_Res\_Offers\_Names(50,80);

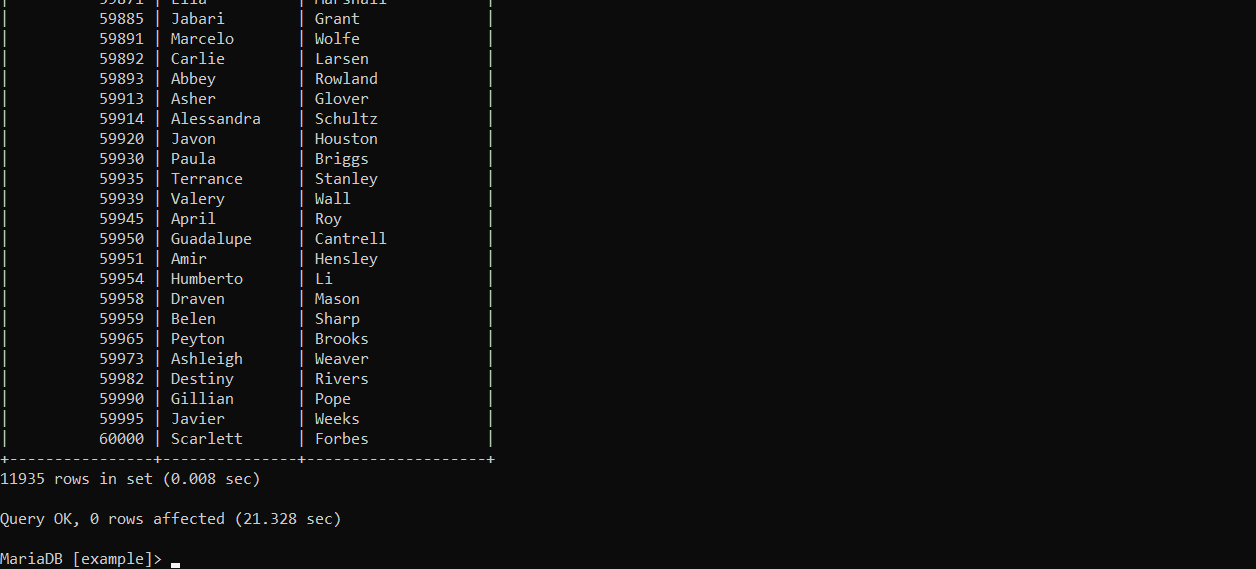
Έξοδος (χωρίς index, παρατίθεται μέρος της εξόδου, λόγω του μεγέθους της):



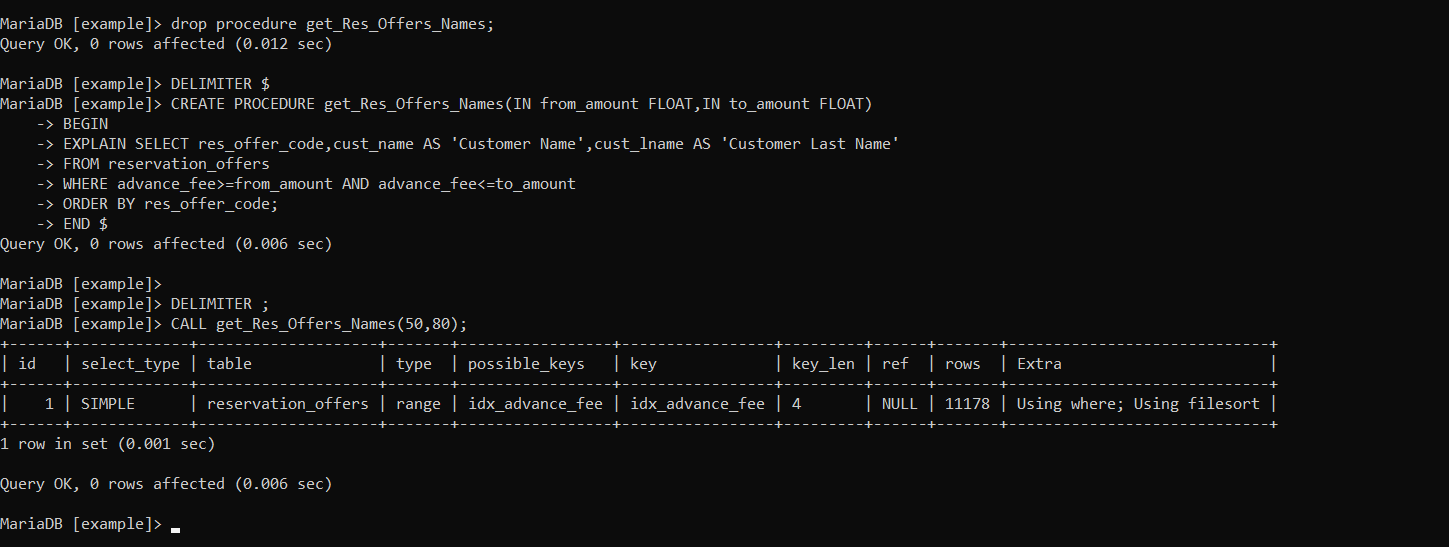
Δημιουργούμε τον index:

-> CREATE INDEX idx\_advance\_fee ON reservation\_offers(advance\_fee) USING BTREE;

Έξοδος (με index):



Παρατηρούμε μια μικρή μείωση του χρόνου εκτέλεσης της τάξης των 3ων δευτερολέπτων, καθώς ίσως το πλήθος των δεδομένων να είναι σχετικά μικρό και να μην μπορεί να φανεί και πολύ μεγάλη διαφορά με ή χωρίς index. Παρόλα αυτά, προσθέτοντας πριν τη ‘SELECT’ εντός της procedure την επιλογή ‘EXPLAIN’ μπορούμε να δούμε ότι κατά την εκτέλεση της εντολής, επιλέγεται να χρησιμοποιηθεί ο index που δηλώσαμε και όχι να προσπελαστεί όλος ο πίνακας:

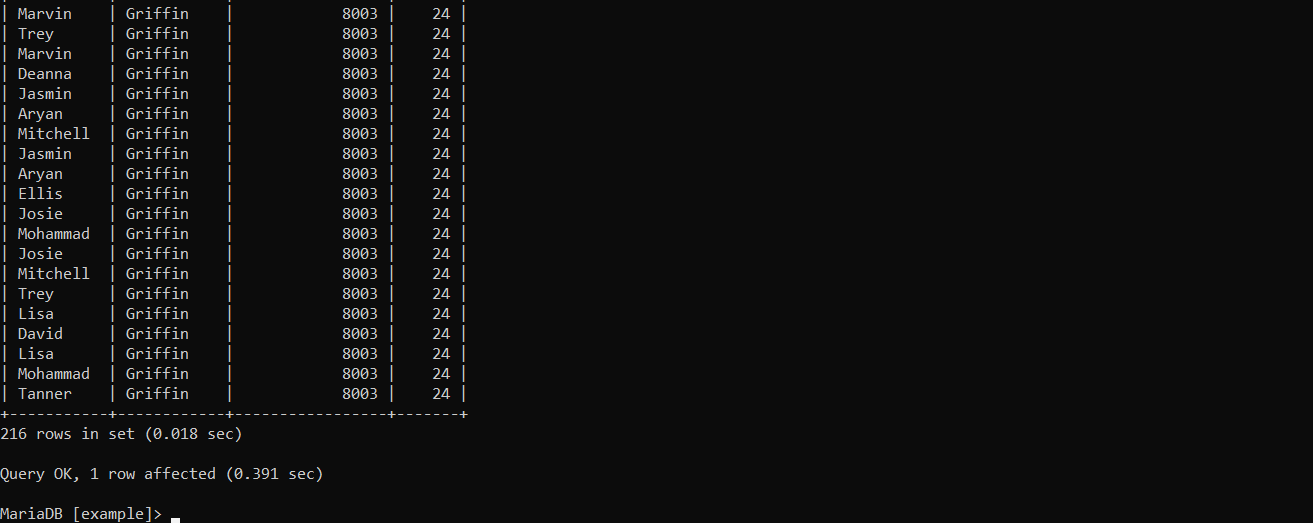


β)

Για την δεύτερη procedure, η αναζήτηση γίνεται με βάση το γνώρισμα cust\_lname, άρα για αυτό θα ορίσουμε τον index στον πίνακα reservation\_offers.

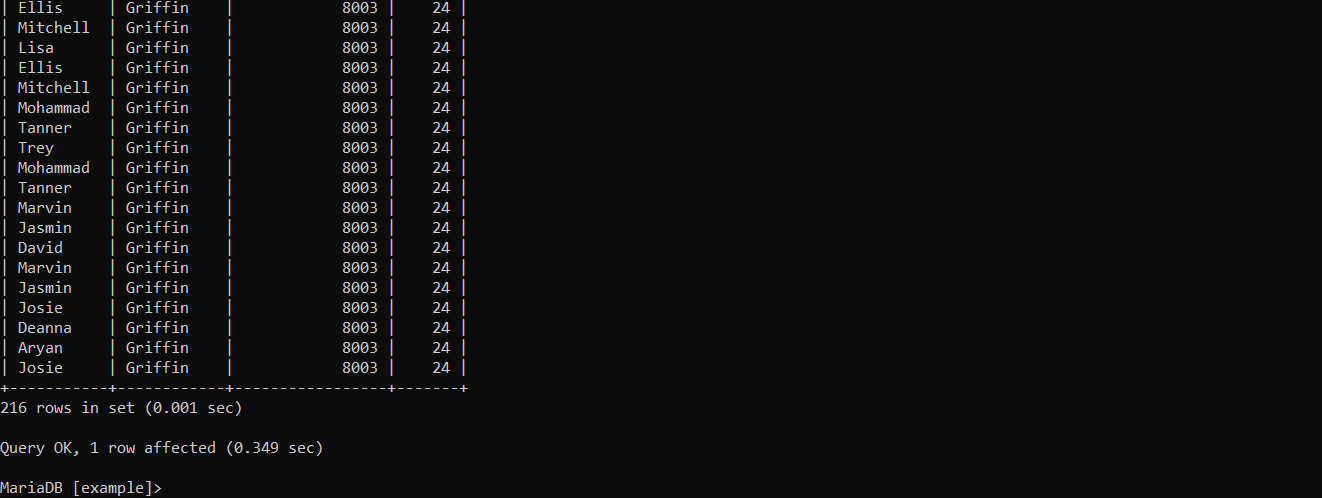
-> CALL get\_Offers\_Participation(‘Griffin’);

Έξοδος (χωρίς index):



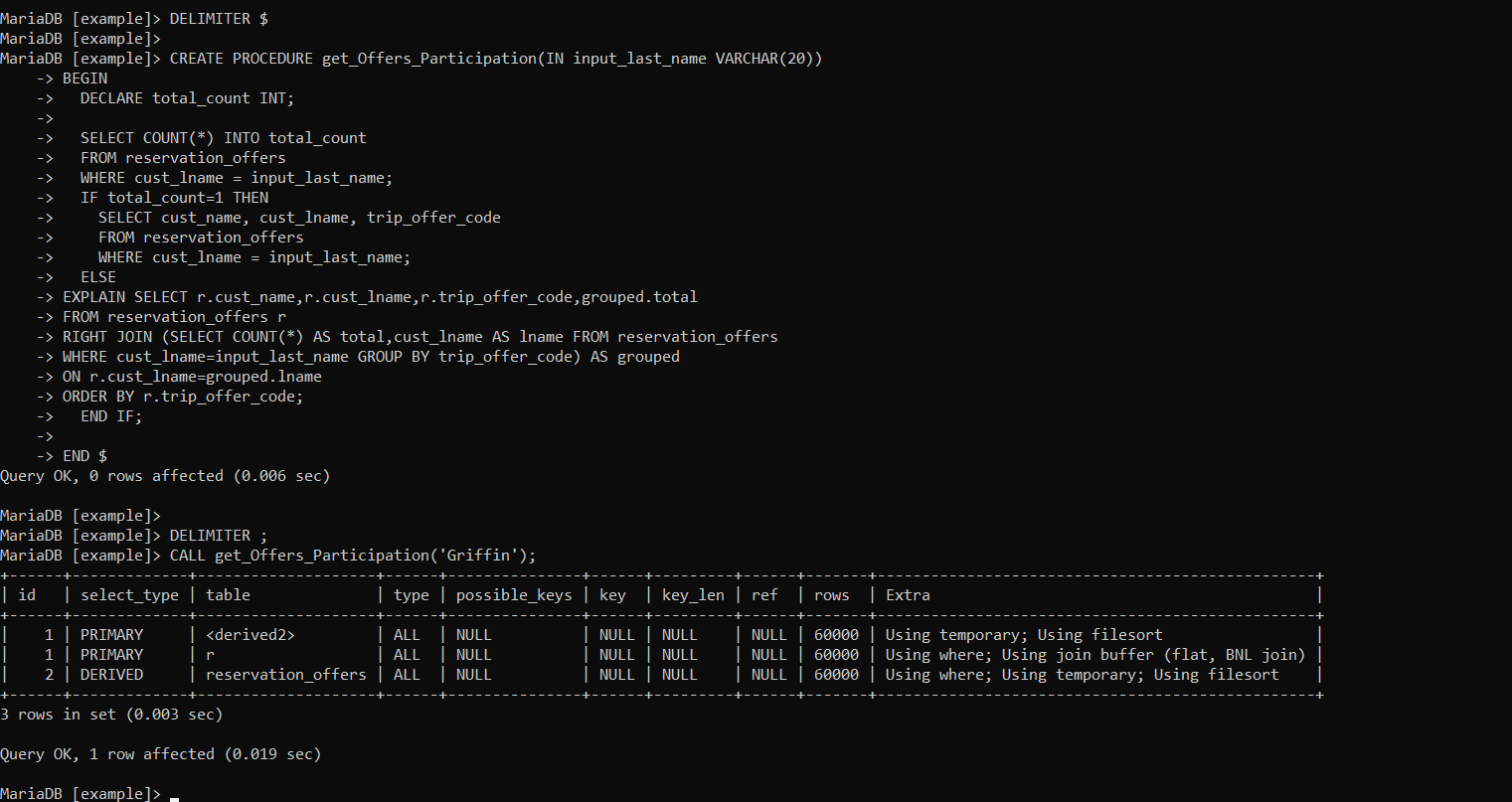
-> CREATE INDEX idx\_cust\_lname ON reservation\_offers(cust\_lname) USING HASH;

Έξοδος (με index):

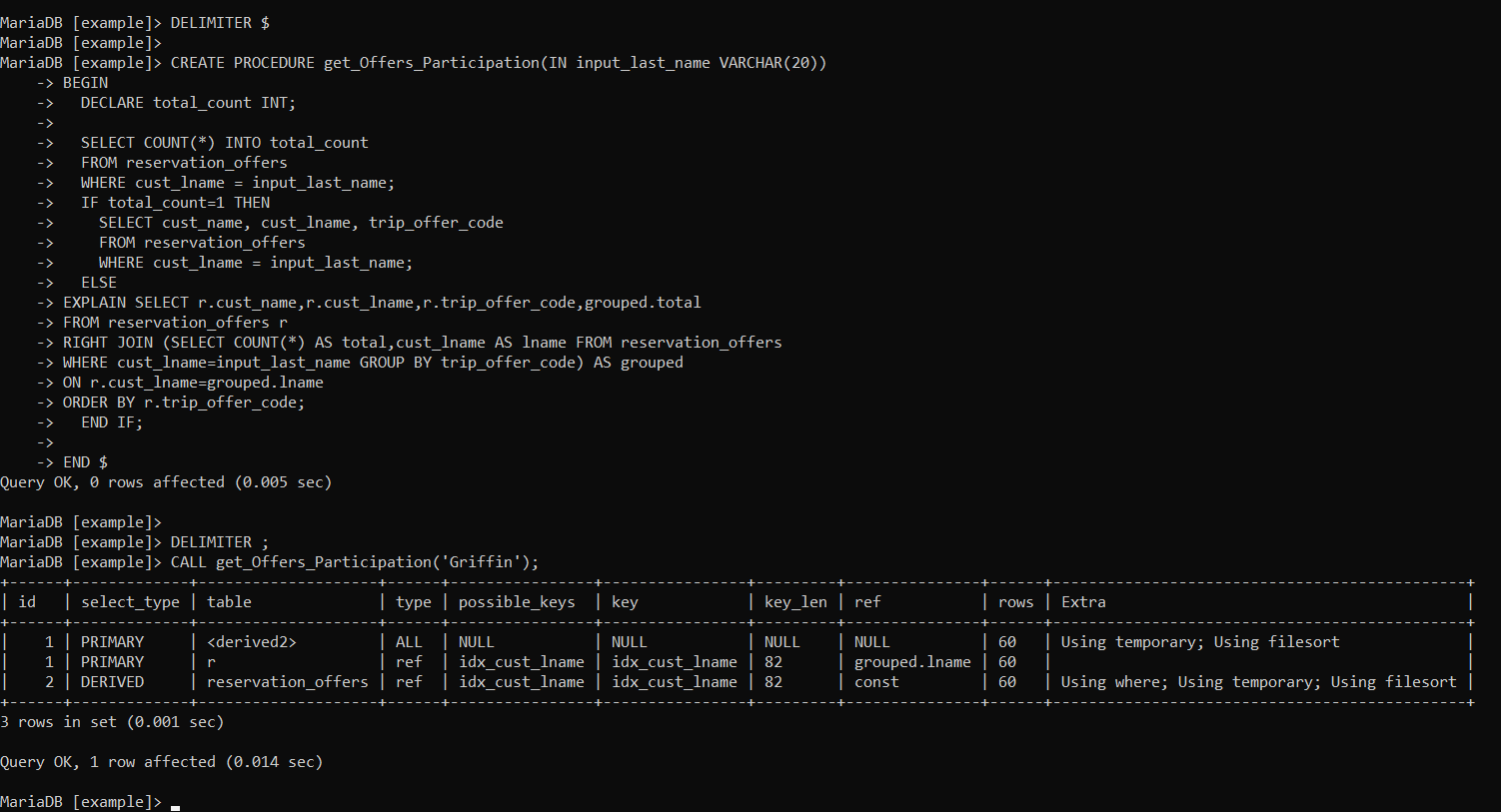


Εδώ δεν παρατηρούμε κάποια διαφορά στον χρόνο εκτέλεσης με ή χωρίς χρήση index. Αυτό μπορεί να οφείλεται και πάλι στο γεγονός ότι το μέγεθος των εγγραφών είναι σχετικά μικρό. Επίσης αφού έχουμε ορίσει Memory ENGINE στον πίνακα, η αναζήτηση θα γίνεται εξ αρχής με χρήση HASH, άρα σε συνδυασμό με το μικρό σχετικά αριθμό εγγραφών να μην μπορούμε να δούμε εμφανή βελτίωση στο χρόνο εκτέλεσης.

Και πάλι με χρήση της ‘EXPLAIN’ και χωρίς χρήση index, βλέπουμε ότι προσπελαύνεται ολόκληρος ο πίνακας κατά την αναζήτηση (στήλη rows: 60.000).



Με χρήση της ‘EXPLAIN’ και με χρήση index, βλέπουμε ότι χρησιμοποιείται, στην δεύτερη και τρίτη ‘SELECT’ ο index και ο αριθμός των γραμμών που πρόκειται να προσπελαστούν είναι κατά πολύ μικρότερος (στήλη rows: 60).



Το πρώτο εμπόδιο που προέκυψε, όταν ορίσαμε το ENGINE = Memory στον πίνακα reservation\_offers, ήταν το σφάλμα: ERROR 1114 (HY000): The table ‘reservation\_offers’ is full.

Αυτό είχε ως αποτέλεσμα να μην μπορούμε να δημιουργήσουμε indexes για τον πίνακα, σε περίπτωση που είχαμε αποθηκεύσει τις εγγραφές πριν τη δημιουργία των indexes, ή να μην μπορεί να αποθηκεύσει όλες τις εγγραφές που θέλαμε (και τις 60.000) και ακόμα να τις διαγράφει όταν κάναμε restart στην βάση.

Αφού είχαμε αρχικοποιήσει τους indexes για τον πίνακα και εισάγαμε σε αυτόν όσα records μπορούσε να πάρει, θέλαμε να δούμε τι χώρο καταλαμβάνουν τα δεδομένα μέχρι στιγμής, ώστε να έχουμε μια εικόνα για το πόση επέκταση θα κάνουμε στο χώρο του πίνακα. Για να δούμε το χώρο που καταλαμβάνει κάθε πίνακας στη βάση δεδομένων μας τρέξαμε την παρακάτω εντολή:

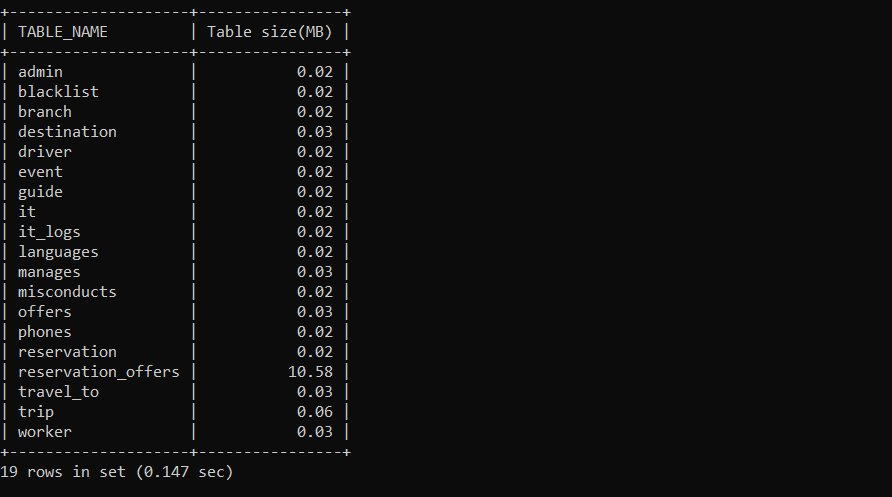
-> SELECT TABLE\_NAME,ROUND(SUM(DATA\_LENGTH + INDEX\_LENGTH) / 1024 / 1024, 2) AS 'Table Size (MB)'

FROM information\_schema.create\_tables

WHERE TABLE\_SCHEMA = 'example'

GROUP BY TABLE\_NAME ORDER BY 'Table Size (MB)' DESC;

Έξοδος:



Προκειμένου να αυξήσουμε τον επιτρεπτό χώρο που μπορεί να καταλαμβάνει κάθε πίνακας με ENGINE = Memory, κάναμε τα εξής βήματα:

1. Ανοίξαμε ένα τερματικό από XAMPP Control Panel και πήγαμε στο φάκελο mysql\bin:

# cd mysql\bin

2. Εκεί βρίσκεται το MariaDB configuration file 'my.ini'. Ανοίγουμε το αρχείο (start notepad my.ini) και προσθέτουμε/τροποποιούμε κάτω από το τμήμα [mysqld] τη μεταβλητή: max\_heap\_table\_size = 64M.

Ορίζουμε ως μέγιστο επιτρεπτό μέγεθος πίνακας δηλαδή τα 64Mb. Η επιλογή των 64Mb είναι λίγο υπερβολική, δεδομένου ότι το ERROR που προκύπτει, λογικά προκύπτει λόγω ανεπάρκειας μερικών Mb, αλλά από

τη στιγμή που δεν υπάρχει κάποια άλλη αυστηρή απαίτηση σε χώρο μπορούμε να κάνουμε αυτή την παραχώρηση.

Μετά από αυτά τα βήματα, ο πίνακας δέχτηκε κανονικά όλες τις εγγραφές που εισάγαμε (και τις 60.000), καθώς και τα indexes.