

# Building A Google Ads Campaign Database with MYSQL: Customer and Campaign Data Tables

Eric Kennedy

In this guide we will be creating data base system using MySQL to store, manage, analyze our clients marketing data . To do this we will installing Linux Ubuntu in a virtual environment. MySQL is what we call a relational data base management system a type of DBS (data base system) that stores data in a structured format. We will visualize this data with Power BI and/or Looker Studio.

## Pre Reqs:

- Any PC or Laptop with at least 8GB of Ram and 256 GB Minimum HD
- Oracle Virtual Box Installed
- Latest Ubuntu ISO file

1. Visit Ubuntu Website to download the latest LTS Ubuntu Version  
<https://ubuntu.com/download/desktop>
2. Instructions to install Ubuntu within Oracle virtual box

[Ubuntu Installation In Virtual Environment \(1\).pdf](#)

5. Open the Terminal in Ubuntu
6. Perform **Repository List Update**: In the terminal type “*sudo apt update*”

## What are Repositories (Linux Ubuntu).(1).pdf

5. If you'd like to see a List of the upgradable repositories:

```
apt list --upgradable
```

### ***Note Random Issue I Encountered:***

You may see that a few repositories won't upgrade and will continue to show as upgradable despite running the update command. This is most likely due to Ubuntu's update phasing.

The message "The following upgrades have been deferred due to phasing" means that some of the packages have been **delayed** from upgrading because they are part of a **phased update**. Phased updates are a method used by Ubuntu to roll out updates gradually to different users in stages to avoid problems with large-scale updates.

### **What is Phasing?**

Phasing is a way for Ubuntu to manage updates safely by controlling how and when updates are delivered to users. Instead of rolling out an update to everyone at once, it's gradually pushed to different groups of users. If your system is in a group that's not yet set to receive the update, it will be deferred until later.

### **What You Can Do:**

#### **Wait for the Update to be Applied Later:**

The easiest thing to do is to

**wait.** Phased updates happen automatically in the background over time. Once your system reaches the point in the phasing process where your updates are scheduled, they will be installed automatically

#### **6. Clean Up After the Upgrade:**

After upgrading, it's a good idea to remove unnecessary packages and clean up the system to free up space. You can do that using the following command:

- `sudo apt autoremove` removes packages that were installed as dependencies but are no longer needed.
- `sudo apt clean` removes downloaded package files that are no longer needed.

## Lets create our own data base

1. Lets install MySQL with a command. In your terminal lets enter a command.

```
sudo apt install mysql-server -y
```

Watch it work its magic as the progress bar below loads to 100%

2. Verify Installation: At this point you are pretty much running MySQL. Let's verify everything is running correctly.

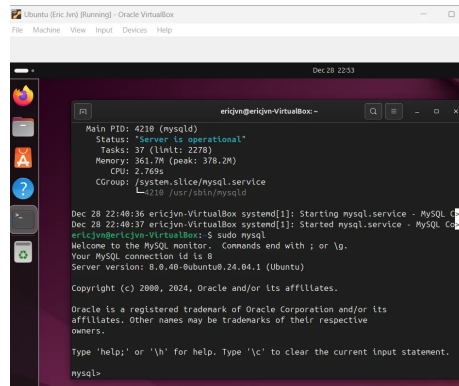
```
sudo systemctl status mysql
```

3. You should see green text that says “active” as well as text that says “server is operational” — type q to get out of there.

## How to access and interact with the databases

1. Type the command below into your terminal

```
sudo mysql
```

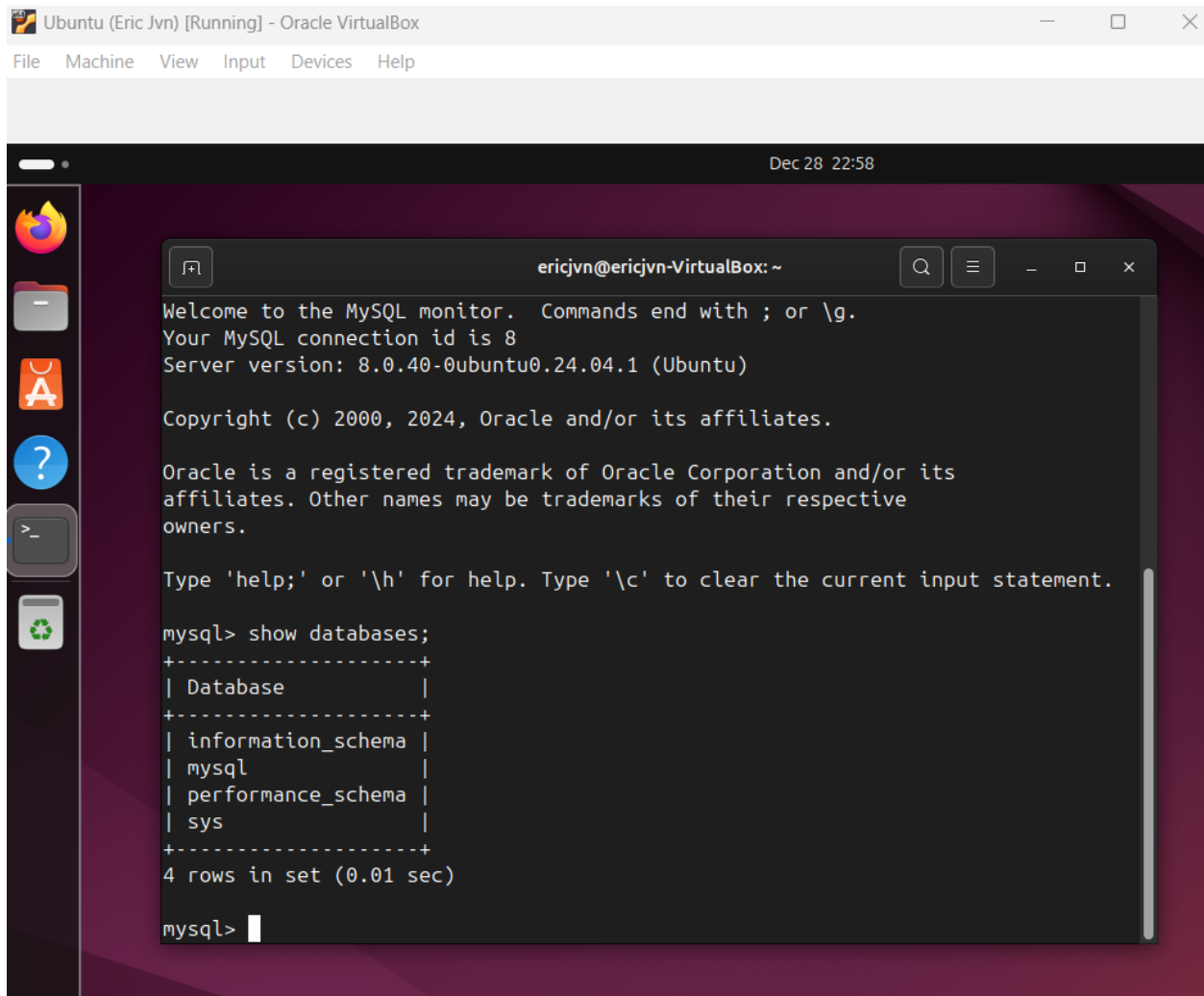


2. You will see at the bottom of the terminal a `mysql>` command prompt appears.

**Note: When typing commands be sure to end them with a semicolon “;” if you dont ..well it just wont work ...so do it!!!**

3. Let view our data base with the command below

```
show databases;
```



We now see some default databases.

## Lets create our own.

1. Create data base

```
create database GoogleAd_Retargeting;
```

Database name: Google Ad Retargeting

As the name suggests this database will contain customer data that our company will use to market and sell their products.

**Note: I personally like to camel case my titles but do know that these letter are case sensitive. Use all lowercase if you prefer.**

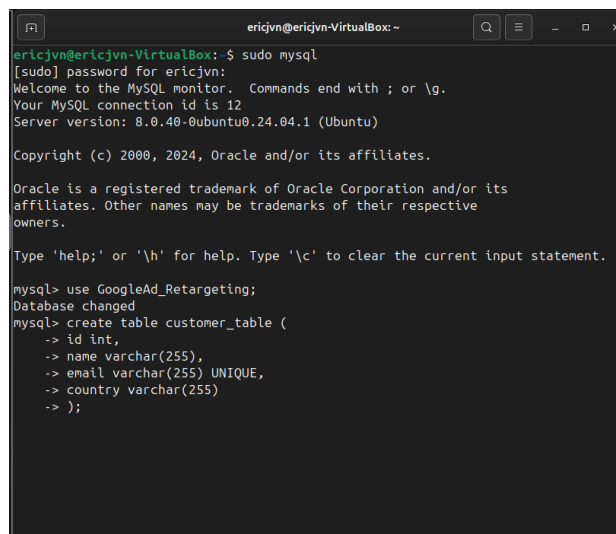
2. Lets switch to our google ad retargeting data base so that way we can begin adding columns and data to it.

```
use GoogleAd_Retargeting;
```

3. Now lets create our first table within our database

Our first table will be the Customer Table which will have our customer data.

4. Type the code in the image below



```
ericjvn@ericjvn-VirtualBox: ~  
ericjvn@ericjvn-VirtualBox:~$ sudo mysql  
[sudo] password for ericjvn:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 12  
Server version: 8.0.40-0ubuntu0.24.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> use GoogleAd_Retargeting;  
Database changed  
mysql> create table customer_table (  
-> id int,  
-> name varchar(255),  
-> email varchar(255) UNIQUE,  
-> country varchar(255)  
-> );
```

Breaking down the Query:

1. int : defines that value to be a integer/number
2. varchar(255) - defines the value as a string and would like to hold the the maximum amount of characters

3. The `UNIQUE` command in SQL is used to ensure that all values in a specific column are distinct or unique across all rows in a table. When you define a column as `UNIQUE`, it enforces a constraint that prevents duplicate values from being inserted into that column.

show the tables

```
show tables;
```

Now lets see what's inside our customer table:

```
describe customer_table;
```

```
ericjvn@ericjvn-VirtualBox: ~  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> use GoogleAd_Retargeting;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_GoogleAd_Retargeting |  
+-----+  
| customer_table                  |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> describe customer_table  
-> \c  
mysql> describe customer_table;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type          | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| id    | int           | YES  |     | NULL    |       |  
| name  | varchar(255)  | YES  |     | NULL    |       |  
| email | varchar(255)  | YES  | UNI | NULL    |       |  
| country | varchar(255) | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
4 rows in set (0.01 sec)  
  
mysql> 
```

When we define our columns, rows and fields..this is apart of defining our SCHEMA and how we will be organizing our data

## Add Row Data

```
INSERT INTO customer_table (id, name, email, country)  
VALUES
```

- (1, 'Xavier Montclair', 'xavier.montclair@vividmail.com', 'France'),
- (2, 'Isabella Storm', 'isabella.storm@skyhunter.org', 'Spain'),
- (3, 'Declan O'Connor', 'declan.oconnor@brightpath.net', 'Ireland'),
- (4, 'Sakura Tanaka', 'sakura.tanaka@dreamwave.jp', 'Japan'),
- (5, 'Diego Velasquez', 'diego.velasquez@tropicalmail.mx', 'Mexico'),
- (6, 'Zoe Winters', 'zoe.winters@frostmail.co.uk', 'United Kingdom');



**View the data:**

```
SELECT * FROM customer_table;
```

```
mysql> select * from customer_table;
+-----+-----+-----+-----+
| id | name | email | country |
+-----+-----+-----+-----+
| 1 | xavier monty | xaviermonty@hotmail.org | france |
| 2 | isabella storm | isabella.storm@skyhunter.org | spain |
| 3 | declan oconnor | docnnor@brightpath.net | united states |
| 4 | sakura uchida | sakurauchida@millenial.io | japan |
| 5 | zoe winters | zoewinters@frostmail.co.uk | united kingdom |
| 6 | marik moto | motozero@moto.io | united states |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

We have now added 6 rows of customer data into our MySQL Database

## Now we will create a Campaigns table

this table will store various information about our google ads campaign such as budget, etc

```
CREATE TABLE campaigns (
  campaign_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT, -- This links to id in customer_table
  campaign_name VARCHAR(255) NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE,
  budget DECIMAL(10, 2) NOT NULL,
  status ENUM('active', 'inactive', 'completed') NOT NULL,
  FOREIGN KEY (customer_id) REFERENCES customer_table(id) -- Creates the foreign key
  link
);
```

**NOTE: Issue: Error 1604 (42000) - Missing Index for Foreign Key**

While working on creating a Google Ads campaign database, I initially created a `customer_table` with basic columns such as `id`, `name`, `email`, and `country`. At this point, I hadn't planned to link the `customer_table` to another table (the `campaigns` table) via a foreign key.

However, later on, I realized that I needed to establish a relationship between the two tables. Specifically, I wanted to link each **campaign** to a **customer**, so I decided to add a **foreign key** to the `campaigns` table. The idea was that the `customer_id` in the `campaigns` table would refer to the `id` column in the `customer_table`, creating a clear relationship between the customer and the campaigns they were associated with.

## Why Link via Foreign Key and Why Index?

- **Foreign Key:** I chose to use a **foreign key** because it ensures referential integrity between the two tables. This means that each campaign must be associated with an existing customer. By using a foreign key, I could prevent errors like linking a campaign to a non-existent customer. This also makes it easier to manage the data, as it enforces constraints that maintain consistency.
- **Indexing:** Foreign keys require the referenced column to be indexed. In this case, the `id` column in the `customer_table` needed to be indexed (either as a **primary key** or **unique key**) because MySQL uses this index to ensure that the foreign key relationship is valid and efficient.

Without this index, MySQL wouldn't be able to enforce the foreign key properly, resulting in an error. The index helps MySQL quickly check if a given `customer_id` in the `campaigns` table matches an `id` in the `customer_table`.

## The Problem I Encountered:

I didn't realize until after creating the `customer_table` that I would need to link the two tables. When I tried to create the `campaigns` table with a foreign key linking to the `id` in `customer_table`, I ran into **Error 1604 (42000)**. This error happened because the `id` column in `customer_table` wasn't indexed, so MySQL couldn't enforce the foreign key relationship.

## How I Resolved It:

1. **Adding the Primary Key:** To fix this, I added a **primary key** to the `id` column in `customer_table`, which automatically created an index for that column:

```
ALTER TABLE customer_table
ADD PRIMARY KEY (id);
```

```
mysql> show index from customer_table;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part |
| Index_type | Comment | Index_comment | Visible | Expression | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| customer_table | 0 | PRIMARY | 1 | id | A | 6 | NULL |
| BTREE | | | YES | NULL | | | |
| customer_table | 0 | email | 1 | email | A | 6 | NULL |
| BTREE | | | YES | NULL | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

You can see the first row that the id is now primary.

1. **Creating the Campaigns Table:** Once the `id` column was indexed, I was able to successfully create the `campaigns` table with the foreign key:

```
CREATE TABLE campaigns (
  campaign_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT, -- This links to `id` in `customer_table`
  campaign_name VARCHAR(255) NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE,
  budget DECIMAL(10, 2) NOT NULL,
  status ENUM('active', 'inactive', 'completed') NOT NULL,
  FOREIGN KEY (customer_id) REFERENCES customer_table(id) -- Creates the foreign
  key link
);
```

## Why This Matters:

- **Data Integrity:** Using the foreign key ensures that every campaign is linked to an existing customer. It prevents situations where a campaign could be added without a valid customer.

- **Efficiency:** Indexing the `id` column in the `customer_table` ensures that the foreign key relationship can be enforced quickly and efficiently, especially as the database grows.

Your SQL query should look like this! Hit enter and you'll get a "query Ok" message meaning that everything went well.

```

ericjvn@ericjvn-VirtualBox: ~
Records: 0 Duplicates: 0 Warnings: 0

mysql> show index form customer_table;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL version for the right syntax to use near 'form customer_table' at line 1
mysql> show index from customer_table
-> ^C
mysql> show index from customer_table;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub |
| Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+
| customer_table | 0 | PRIMARY | 1 | id | A | 6 |
| BTREE | | | YES | NULL |
| customer_table | 0 | email | 1 | email | A | 6 |
| BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> create table campaigns_table (
-> campaign_sid int auto_increment primary key,
-> customer_id int,
-> campaign_name varchar(255) not null,
-> start_date date not null,
-> end_date date,
-> budget decimal(10,2) not null,
-> status ENUM('active', 'inactive', 'completed') not null,
-> FOREIGN KEY (customer_id) REFERENCES customer_table(id)
-> );

```

Lets verify

show the tables

```
show tables;
```

Now lets see what's inside our customer table:

```
describe campaigns_table;
```

Looks liek i made a typo in the 1st field. it should say

campaigns\_id but instead says campaign\_sid

lets fix this

## Steps to Edit a Column in MySQL:

1. **Identify the column** with the typo.
2. Use the `ALTER TABLE` command to rename the column or modify its data type, length, or other properties.

## Common Scenarios:

### 1. Rename a Column (Correcting Typo in Column Name):

If you made a typo in the **column name**, you can rename the column with the `CHANGE COLUMN` command.

Syntax:

For example, if you accidentally named a column `custmer_name` instead of `customer_name`, you would do the following:

```
ALTER TABLE customer_table
CHANGE COLUMN custmer_name customer_name VARCHAR(255);
```

This will change the column name from `custmer_name` to `customer_name` while keeping the data type as `VARCHAR(255)`.

```
mysql> describe campaigns_table;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| campaign_sid | int           | NO   | PRI | NULL    | auto_increment |
| customer_id  | int           | YES  | MUL | NULL    |                |
| campaign_name | varchar(255)  | NO   |     | NULL    |                |
| start_date   | date          | NO   |     | NULL    |                |
| end_date     | date          | YES  |     | NULL    |                |
| budget       | decimal(10,2) | NO   |     | NULL    |                |
| status       | enum('active','inactive','completed') | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> alter table campaigns_table
-> change column campaign_sid campaigns_id varchar(255);
```

now press enter

```
mysql> alter table campaigns_table
-> change column campaign_sid campaigns_id varchar(255);
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> █
```

Query Ok

**Now lets verify updates**

```
describe campaigns_table;
```

```
mysql> describe campaigns_table;
+-----+-----+-----+-----+-----+-----+
| Field          | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| campaigns_id   | varchar(255)        | NO   | PRI | NULL    |       |
| customer_id    | int                 | YES  | MUL | NULL    |       |
| campaign_name  | varchar(255)        | NO   |     | NULL    |       |
| start_date     | date                | NO   |     | NULL    |       |
| end_date       | date                | YES  |     | NULL    |       |
| budget         | decimal(10,2)       | NO   |     | NULL    |       |
| status         | enum('active','inactive','completed') | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> █
```

**Now lets add our rows.**

```
INSERT INTO campaigns_table (customer_id, campaign_name, start_date, end_date, budget,
status)
```

```
VALUES
```

```
(1, 'Winter Sale 2024', '2024-01-01', '2024-02-01', 5000.00, 'active'),
(2, 'New Year Promotion', '2024-01-05', '2024-01-20', 1500.00, 'completed'),
(3, 'Summer Collection', '2024-06-01', '2024-06-30', 7000.00, 'active'),
(4, 'Black Friday Deals', '2024-11-01', '2024-11-30', 10000.00, 'inactive'),
(5, 'Christmas Campaign', '2024-12-01', '2024-12-25', 12000.00, 'active'),
(6, 'Spring Clearance', '2024-03-01', '2024-03-31', 4000.00, 'inactive');
```

Note: If you have a csv file that is properly formatted , you can use this SQL query to import the csv into your table. We won't go into depth on how to do this so if you run into any errors just manually enter the data as we have shown you how to do before. We will do more projects with larger sets of data where importing may be necessary.

```
LOAD DATA INFILE '/path/to/campaigns.csv'
INTO TABLE campaigns_table
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Error we ran into: Realized that I assigned the campaigns\_id field as a var(255) instead of an INT. Updated this.

- , I didn't follow best practices for designing primary keys. Numeric values ( `INT` or `BIGINT` ) are generally preferred for such columns because they are more efficient to store and search.

## How I resolved this:

1. Change `campaign_id` to `INT` (or `BIGINT` if you need larger numbers) with `AUTO_INCREMENT` to automatically generate unique IDs. This is how it should be structured:

```
ALTER TABLE campaigns MODIFY campaign_id INT AUTO_INCREMENT;
```

Now after the data is entered your campaigns table should look like this

```
ericjvn@ericjvn-VirtualBox: ~  
ERROR 1146 (42S02): Table 'GoogleAd_Retargeting.camapigns_table' doesn't exist  
mysql> select * from campaigns_table  
-> ^C  
mysql> select * from campaigns_table;  
+-----+-----+-----+-----+-----+-----+  
| campaigns_id | customer_id | campaign_name | start_date | end_date | budget |  
+-----+-----+-----+-----+-----+-----+  
| 1 | 1 | Winter Sale 2024 | 2024-01-01 | 2024-02-01 | 5000.00 |  
| 2 | 2 | New Year Promotion | 2024-01-05 | 2024-01-20 | 1500.00 |  
| 3 | 3 | Summer Collection | 2024-06-01 | 2024-06-30 | 7000.00 |  
| 4 | 4 | Black Friday Deals | 2024-11-01 | 2024-11-30 | 10000.00 |  
| 5 | 5 | Christmas Campaign | 2024-12-01 | 2024-12-25 | 12000.00 |  
| 6 | 6 | Spring Clearance | 2024-03-01 | 2024-03-31 | 4000.00 |  
+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

I have successfully created a Google Ads campaign database, which includes two tables: one for storing customer data and another for campaign data.

## Building a Google Ads Campaign Database with MySQL: Customer and Campaign Data Tables