OneCompiler

EDITOR CHALLENGES ORGS MORE LOGIN

queries.sql

42pu2dmue

NEW MYSQL RUN

```sql

CREATE TABLE clientes (
id INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(100) NOT NULL, email VARCHAR (100) UNIQUE NOT
);

CREATE TABLE produtos (
id INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR (100) NOT NULL, preco DECIMAL (10, 2) NOT NUL
);

CREATE TABLE fornecedores (
id INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR (100) NOT NULL, telefone VARCHAR (20) NOT NUL
CREATE TABLE funcionarios (
id INT AUTO_
INCREMENT PRIMARY KEY,
nome VARCHAR (100) NOT NULL, cargo VARCHAR (50) NOT NULL
INSERT INTO clientes (nome, email)
VALUES ('',henrique 'rickappereira@gmail.com');
INSERT INTO produtos (nome, preco)
VALUES ('charrete', 13000.00);
INSERT INTO clientes (nome, email)
VALUES (henrique', rickappereira@gmail.com ');
INSERT INTO produtos (nome, preco)
VALUES ('charrete', 13000.00);
INSERT INTO fornecedores (nome, telefone)
VALUES ('Fornecedor x', '1234-5678');
INSERT INTO funcionarios (nome, cargo)
VALUES ('Carlos'
, 'Analista');
SELECT * FROM clientes
WHERE email = rickappereira@gmail.com';
```

STDIN

Input for the program (Optional)

Output:

```
ERROR 1064 (42000) at line 10: You have an error in your SQL
id INT AUTO_
INCREMENT PRIMARY KEY,
nome VARCHAR (10' at line 3
```

FlaiChat breaks language barriers with Automatic Translations. Available on iOS and Android.
SPONSORED

MySQL online editor

Write, Run & Share MySQL queries online using OneCompiler's MySQL online editor and compiler for free. It's one of the robust, feature-rich online editor and compiler for MySQL. Getting started with the OneCompiler's MySQL editor is really simple and pretty fast. The editor shows sample boilerplate code when you choose language as 'MySQL' and start writing queries to learn and test online without worrying about tedious process of installation.

About MySQL

MySQL is a open-source, free and very popular relational database management system which is developed, distributed and supported by Oracle corporation.

Key Features:

- Open-source relational database management systems.
- Reliable, very fast and easy to use database server.
- Works on client-server model.
- Highly Secure and Scalable
- High Performance
- High productivity as it uses stored procedures, triggers, views to write a highly productive code.
- Supports large databases efficiently.
- Supports many operating systems like Linux*,CentOS*, Solaris*,Ubuntu*,Windows*, MacOS*,FreeBSD* and others.

Syntax help

Commands

1. CREATE

```
CREATE TABLE table_name (
                column1 datatype,
                column2 datatype,
                ....);
```

Example

```
CREATE TABLE EMPLOYEE (
  empId INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  dept TEXT NOT NULL
);
```

2. ALTER

```
ALTER TABLE Table_name ADD column_name datatype;
```

Example

```
INSERT INTO EMPLOYEE VALUES (0001, 'Dave', 'Sales');
```

3. TRUNCATE

```
TRUNCATE table table_name;
```

4. DROP

```
DROP TABLE table_name;
```

5. RENAME

```
RENAME TABLE table_name1 to new_table_name1;
```

6. COMMENT

Single-Line Comments:

```
 --Line1;
```

Multi-Line comments:

```
   /* Line1,
   Line2 */
```

DML Commands

1. INSERT

```
INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);
```

Note: Column names are optional.

Example

```
INSERT INTO EMPLOYEE VALUES (0001, 'Ava', 'Sales');
```

2. SELECT

```
SELECT column1, column2, ...
FROM table_name
[where condition];
```

Example

```
SELECT * FROM EMPLOYEE where dept ='sales';
```

3. UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Example

```
UPDATE EMPLOYEE SET dept = 'Sales' WHERE empId='0001';
```

4. DELETE

```
DELETE FROM table_name where condition;
```

Example

```
DELETE from EMPLOYEE where empId='0001';
```

Indexes

1. CREATE INDEX

```
  CREATE INDEX index_name on table_name(column_name);
```

- To Create Unique index:

```
  CREATE UNIQUE INDEX index_name on table_name(column_name);
```

2. DROP INDEX

```
DROP INDEX index_name ON table_name;
```

Views

1. Create a View

```
Creating a View:
CREATE VIEW View_name AS
Query;
```

2. How to call view

```
SELECT * FROM View_name;
```

3. Altering a View

```
ALTER View View_name AS
Query;
```

4. Deleting a View

```
DROP VIEW View_name;
```

Triggers

1. Create a Trigger

```
CREATE TRIGGER trigger_name trigger_time trigger_event
    ON tbl_name FOR EACH ROW [trigger_order] trigger_body
/* where
trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE | DELETE }
trigger_order: { FOLLOWS | PRECEDES } */
```

2. Drop a Trigger

```
DROP TRIGGER [IF EXISTS] trigger_name;
```

Stored Procedures

1. Create a Stored Procedure

```
CREATE PROCEDURE sp_name(p1 datatype)
BEGIN
/*Stored procedure code*/
END;
```

2. How to call Stored procedure

```
CALL sp_name;
```

3. How to delete stored procedure

```
DROP PROCEDURE sp_name;
```

Joins

1. INNER JOIN

```
SELECT * FROM TABLE1 INNER JOIN TABLE2 where condition;
```

2. LEFT JOIN

```
SELECT * FROM TABLE1 LEFT JOIN TABLE2 ON condition;
```

3. RIGHT JOIN

```
SELECT * FROM TABLE1 RIGHT JOIN TABLE2 ON condition;
```

4. CROSS JOIN

```
SELECT select_list from TABLE1 CROSS JOIN TABLE2;
```