

**PROJECT BASED LEARNING**

**PROJECT REPORT SUBMITTED IN A**

**SEMESTER 5 OF BACHELOR OF TECHNOLOGY**

**IN COMPUTER SCIENCE AND ENGINEERING BY**

**21331A0530 – Ch.Yashwanth**

**21331A0549 – G.Anusha**

**21331A0556 – G.Varshit Varma**

**21331A0557– G.Sai Hemanth Kumar**

**21331A0559 – G.Amani**

Under the esteemed guidance of

**Mr. M.Vamsi Krishna**

**Department of Computer Science and Engineering**

**MVGR College of Engineering**

**Table of Contents :**

[**1.Introduction 2**](#_dhde49cjtub8)

[**2.Project Overviews 2**](#_2w1ujjbvq7jx)

[**3.List of Entities & Attributes 2**](#_6n7y9n1a0aml)

[**4.List of Relationships 3**](#_2gda5il3q1q7)

[**5.Basics To Do 3**](#_rf1rpz8lzf8e)

[5.1.Design the logical view using ER Diagrams with tools 3](#_jaf434todd12)

[5.2.Design Enhanced ER diagram using Workbench 4](#_3gv660qdl78l)

[5.3.Forward Engineer your EER diagrams in Workbench 5](#_27msyfyumbr5)

[5.4.SQL Queries to demonstrate the working 10](#_aavri0cktrx3)

[5.4.1.Select Query 10](#_5zy1hzhq39be)

[5.4.2.Insert Query 11](#_ds3p6dtaem5f)

[**6.Actual Tables in Database 13**](#_olnu5r1uclms)

[6.1.Category 13](#_2lo96crk5fw1)

[6.2.Event 14](#_1tczb5gx7q45)

[6.3.Post 14](#_c0kiiqyjvo9w)

[6.4.Comment 15](#_h0s3kfpkj2i4)

[6.5.Tag 16](#_q4a1y3j7qwo3)

[6.6 Post\_has\_Tag 17](#_rves13flw7ok)

[**7.Implement SQL Queries to display in 18**](#_cgdhkgysba2s)

[7.1.Mysql Workbench(using Mysql) 18](#_7xy3a7riju9n)

[7.1.1.Popular blog posts 18](#_6ekpvg1hl838)

[7.1.2.Manage comments 18](#_j8jkn03mbh9i)

[7.1.3.Categorize posts 19](#_utpcwo4dtvwr)

[7.2.IDLE & Visual Studio Code(using Python) 19](#_s1qe6j3u6jum)

[**8.Remarks 21**](#_ok5neq5mch4q)

# 

# **1.Introduction**

This is a project on a “Blogging Platform” where we have Blog Posts, Blog Categories, Tags of a Blog, Comments on the posts of a Blog. We need to manage all the data of the blogs in an efficient way where we can store the data in an efficient way with reduction of duplicate values in the tables and easy to handle the data. We need to organise the data and provide users more access and data control over their data.

# **2.Project Overviews**

The Blogging Platform Project in DBMS revolves around the creation of a robust and user-friendly platform where individuals and entities can create and manage blog posts, categorise them, receive feedback through comments, enhance discoverability with tags, and organise or participate in events. The primary focus of this project is the design and implementation of a sophisticated database system that efficiently handles the diverse data associated with these entities.

# **3.List of Entities & Attributes**

| **Entity** | **Attribute** |
| --- | --- |
| Category | Category\_ID, Category\_Type |
| Event | Event\_ID, Event\_Name, Event\_Organiser\_Name |
| Post | Post\_ID, Post\_Article, Post\_Views |
| Comment | Cmt\_ID, Cmt\_Data |
| Tag | Tag\_ID, Tag\_Word |

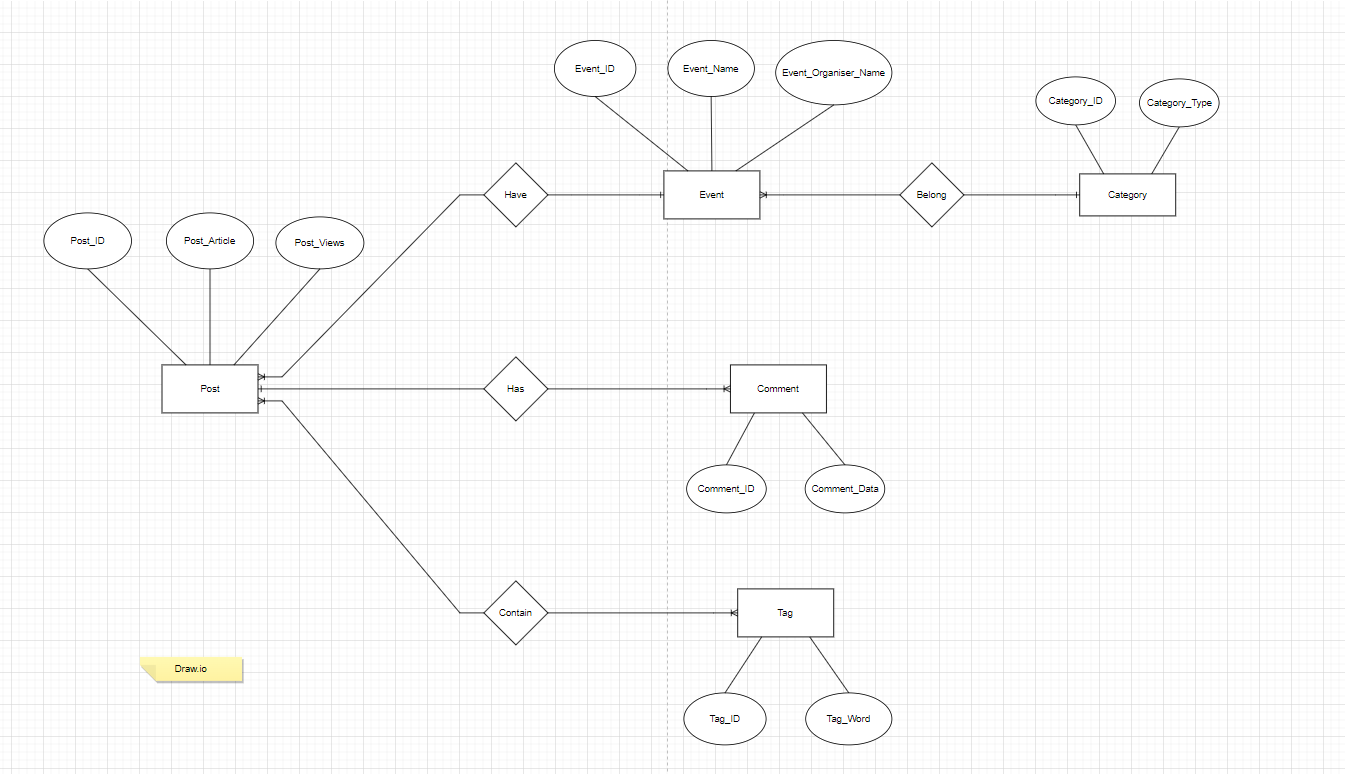
# **4.List of Relationships**

* One or many Events can belong to one Category.
* One Event can have many Posts.
* One or many Posts can contain one or many Tags.
* One Post can have many Comments.

# **5.Basics To Do**

## **5.1.Design the logical view using ER Diagrams with tools**

ER Diagram :

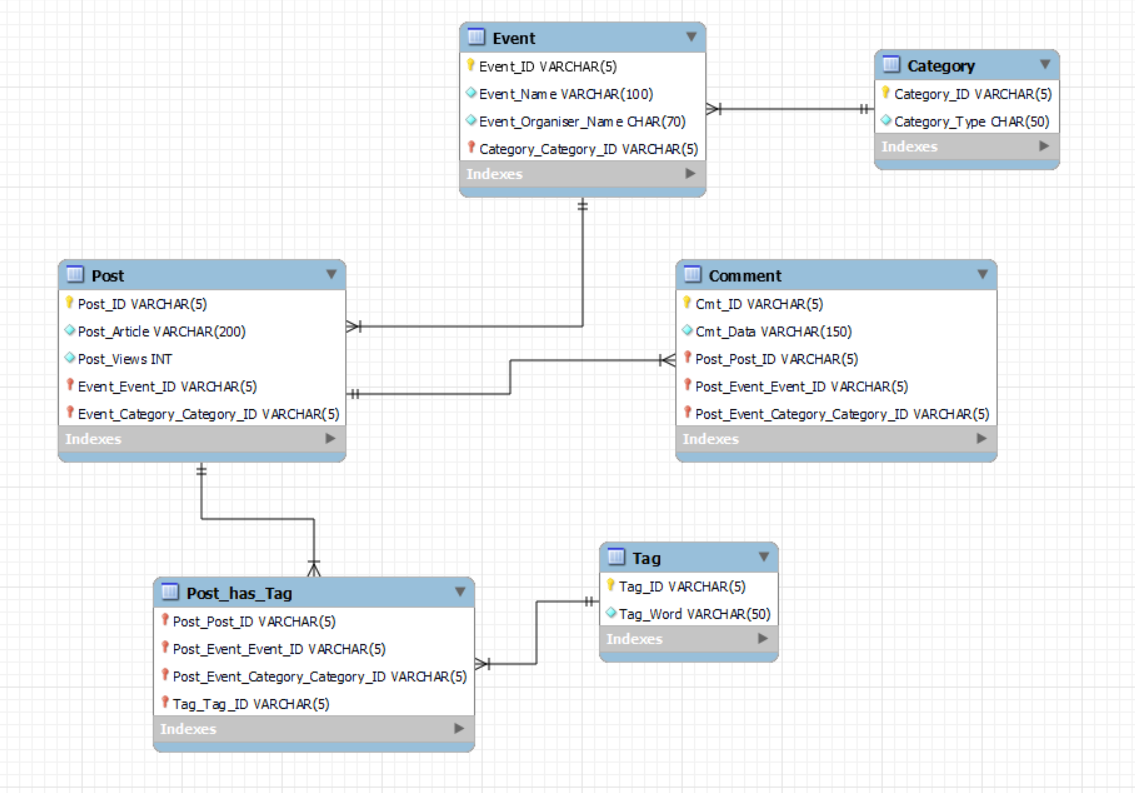


## 

## 

## **5.2.Design Enhanced ER diagram using Workbench**

EER Diagram :

****

## **5.3.Forward Engineer your EER diagrams in Workbench**

Schema :

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`Category`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Category` (

`Category\_ID` VARCHAR(5) NOT NULL,

`Category\_Type` CHAR(50) NOT NULL,

PRIMARY KEY (`Category\_ID`),

UNIQUE INDEX `Category\_ID\_UNIQUE` (`Category\_ID` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Event`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Event` (

`Event\_ID` VARCHAR(5) NOT NULL,

`Event\_Name` VARCHAR(100) NOT NULL,

`Event\_Organiser\_Name` CHAR(70) NOT NULL,

`Category\_Category\_ID` VARCHAR(5) NOT NULL,

PRIMARY KEY (`Event\_ID`, `Category\_Category\_ID`),

UNIQUE INDEX `Event\_ID\_UNIQUE` (`Event\_ID` ASC) VISIBLE,

INDEX `fk\_Event\_Category\_idx` (`Category\_Category\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Event\_Category`

FOREIGN KEY (`Category\_Category\_ID`)

REFERENCES `mydb`.`Category` (`Category\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Post`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Post` (

`Post\_ID` VARCHAR(5) NOT NULL,

`Post\_Article` VARCHAR(600) NOT NULL,

`Post\_Views` INT NOT NULL,

`Event\_Event\_ID` VARCHAR(5) NOT NULL,

`Event\_Category\_Category\_ID` VARCHAR(5) NOT NULL,

PRIMARY KEY (`Post\_ID`, `Event\_Event\_ID`, `Event\_Category\_Category\_ID`),

UNIQUE INDEX `Post\_ID\_UNIQUE` (`Post\_ID` ASC) VISIBLE,

INDEX `fk\_Post\_Event1\_idx` (`Event\_Event\_ID` ASC, `Event\_Category\_Category\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Post\_Event1`

FOREIGN KEY (`Event\_Event\_ID` , `Event\_Category\_Category\_ID`)

REFERENCES `mydb`.`Event` (`Event\_ID` , `Category\_Category\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Comment`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Comment` (

`Comment\_ID` VARCHAR(5) NOT NULL,

`Comment\_Data` VARCHAR(500) NOT NULL,

`Post\_Post\_ID` VARCHAR(5) NOT NULL,

`Post\_Event\_Event\_ID` VARCHAR(5) NOT NULL,

`Post\_Event\_Category\_Category\_ID` VARCHAR(5) NOT NULL,

PRIMARY KEY (`Comment\_ID`, `Post\_Post\_ID`, `Post\_Event\_Event\_ID`, `Post\_Event\_Category\_Category\_ID`),

UNIQUE INDEX `Comment\_ID\_UNIQUE` (`Comment\_ID` ASC) VISIBLE,

INDEX `fk\_Comment\_Post1\_idx` (`Post\_Post\_ID` ASC, `Post\_Event\_Event\_ID` ASC, `Post\_Event\_Category\_Category\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Comment\_Post1`

FOREIGN KEY (`Post\_Post\_ID` , `Post\_Event\_Event\_ID` , `Post\_Event\_Category\_Category\_ID`)

REFERENCES `mydb`.`Post` (`Post\_ID` , `Event\_Event\_ID` , `Event\_Category\_Category\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Tag`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Tag` (

`Tag\_ID` VARCHAR(5) NOT NULL,

`Tag\_Word` VARCHAR(50) NOT NULL,

PRIMARY KEY (`Tag\_ID`),

UNIQUE INDEX `Tag\_ID\_UNIQUE` (`Tag\_ID` ASC) VISIBLE,

UNIQUE INDEX `Tag\_Word\_UNIQUE` (`Tag\_Word` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Post\_has\_Tag`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Post\_has\_Tag` (

`Post\_Post\_ID` VARCHAR(5) NOT NULL,

`Post\_Event\_Event\_ID` VARCHAR(5) NOT NULL,

`Post\_Event\_Category\_Category\_ID` VARCHAR(5) NOT NULL,

`Tag\_Tag\_ID` VARCHAR(5) NOT NULL,

PRIMARY KEY (`Post\_Post\_ID`, `Post\_Event\_Event\_ID`, `Post\_Event\_Category\_Category\_ID`, `Tag\_Tag\_ID`),

INDEX `fk\_Post\_has\_Tag\_Tag1\_idx` (`Tag\_Tag\_ID` ASC) VISIBLE,

INDEX `fk\_Post\_has\_Tag\_Post1\_idx` (`Post\_Post\_ID` ASC, `Post\_Event\_Event\_ID` ASC, `Post\_Event\_Category\_Category\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Post\_has\_Tag\_Post1`

FOREIGN KEY (`Post\_Post\_ID` , `Post\_Event\_Event\_ID` , `Post\_Event\_Category\_Category\_ID`)

REFERENCES `mydb`.`Post` (`Post\_ID` , `Event\_Event\_ID` , `Event\_Category\_Category\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Post\_has\_Tag\_Tag1`

FOREIGN KEY (`Tag\_Tag\_ID`)

REFERENCES `mydb`.`Tag` (`Tag\_ID`)

ON DELETE NO ACTION

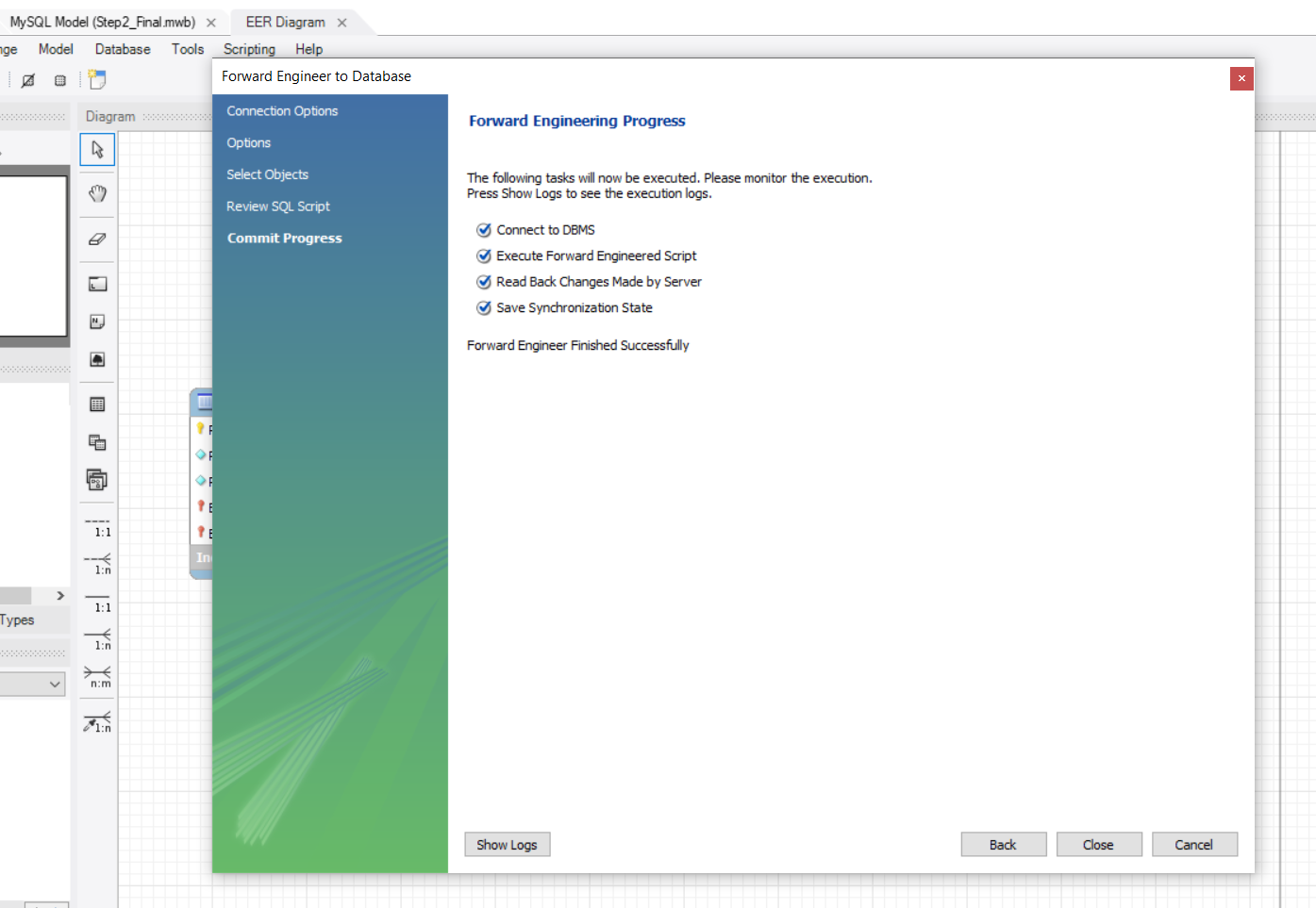
ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;



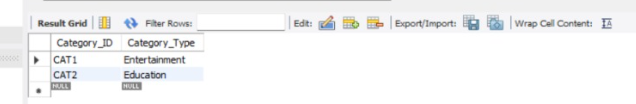
## 

## **5.4.SQL Queries to demonstrate the working**

### **5.4.1.Select Query**

Original table :

Category Table :



Query :

import mysql.connector as c

from tabulate import tabulate

conn = c.connect(

user='Pbl',

password='Pbl@123',

host='localhost',

database='mydb'

)

cursor = conn.cursor()

sql1 = 'SELECT \* FROM Category'

cursor.execute(sql1)

result1 = cursor.fetchall()

def print\_results\_as\_table(query, results):

print("Query:", query)

headers = [i[0] for i in cursor.description]

print(tabulate(results, headers=headers, tablefmt="fancy\_grid"))

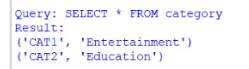
print()

print\_results\_as\_table(sql1, result1)

cursor.close()

conn.close()

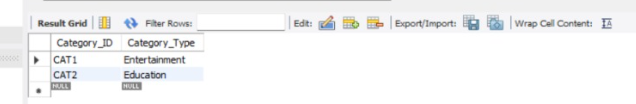
Output :



### **5.4.2.Insert Query**

Original table :

Category Table :



Query **:**

import mysql.connector as c

from tabulate import tabulate

from termcolor import colored

conn = c.connect(user='Pbl', password='Pbl@123', host='localhost', database='mydb')

cursor = conn.cursor()

# Display the Category table before the update

cursor.execute("SELECT \* FROM Category;")

category\_result, headers = cursor.fetchall(), [colored(i[0], "blue") for i in cursor.description]

table\_data = [[colored(cell, 'red') if cell in ['CAT3', 'Spiritual'] else colored(cell, 'blue') for cell in row] for row in category\_result]

print(tabulate(table\_data, headers=headers, tablefmt="fancy\_grid"))

# Insert a new record into the Category table

insert\_query = "INSERT INTO Category (Category\_ID, Category\_Type) VALUES (%s, %s)"

values = ('CAT3', 'Spiritual')

try:

cursor.execute(insert\_query, values)

conn.commit()

print(colored("Record added successfully:", "green"))

except c.Error as err:

print(colored(f"Error: {err}", "red"))

# Display the Category table after the update

cursor.execute("SELECT \* FROM Category;")

category\_result, headers = cursor.fetchall(), [colored(i[0], "blue") for i in cursor.description]

table\_data = [[colored(cell, 'red') if cell in ['CAT3', 'Spiritual'] else colored(cell, 'blue') for cell in row] for row in category\_result]

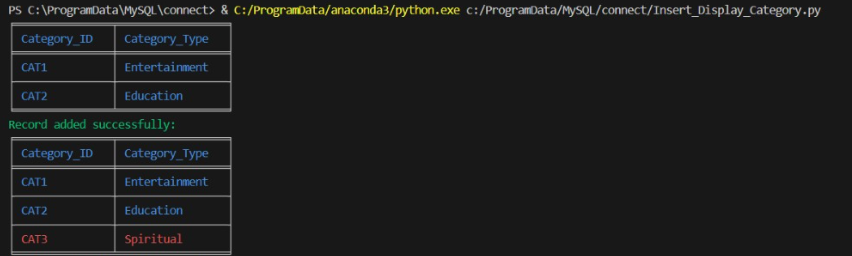
print(tabulate(table\_data, headers=headers, tablefmt="fancy\_grid"))

# Close the cursor and connection

cursor.close()

conn.close()

Output :



# **6.Actual Tables in Database**

## **6.1.Category**

Query :

-- category Values--

insert into Category (Category\_ID, Category\_Type)

values

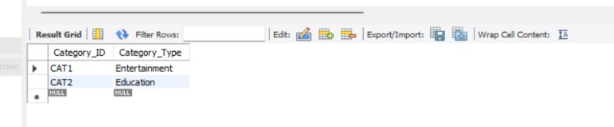
('CAT1','Entertainment'),

('CAT2', 'Education');

select \* from Category;

drop table Category;

Output :



## **6.2.Event**

Query :

-- Events Values--

INSERT INTO events (Event\_ID, Event\_Name, Event\_Organiser\_Name, Category\_Category\_ID)

VALUES

('EV1', 'Silver Jublee', 'Anusha', 'CAT1'),

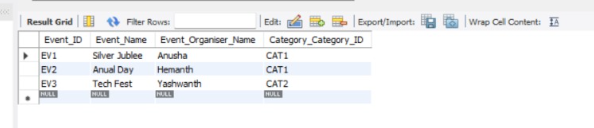
('EV2', 'Annual Day', 'Hemanth', 'CAT1'),

('EV3', 'Tech Fest', 'Yashwanth', 'CAT2');

select \* from Event;

drop table Event;

Output :



## **6.3.Post**

Query :

-- values in Post --

insert into post values

("PT1","Article\_A",1000,"EV3","CAT2"),

("PT2","Article\_B",530,"EV3","CAT2"),

("PT3","Article\_C",500,"EV1","CAT1"),

("PT4","Article\_D",557,"EV2","CAT1"),

("PT5","Article\_E",559,"EV2","CAT1"),

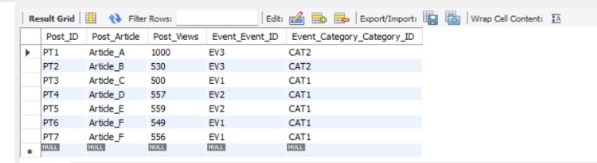
("PT6","Article\_F",549,"EV1","CAT1"),

("PT7","Article\_F",556,"EV1","CAT1");

select \* from Post;

drop table post;

Output :



## **6.4.Comment**

Query :

-- values in Comment --

insert into Comment values

("COM1","Nice","PT6","EV1","CAT1"),

("COM2","Average","PT4","EV2","CAT1"),

("COM3","Interesting","PT2","EV3","CAT2"),

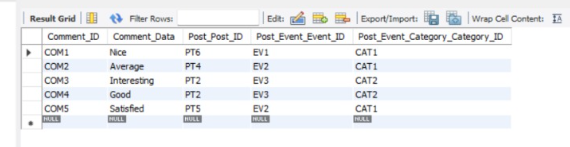
("COM4","Good","PT2","EV3","CAT2"),

("COM5","Satisfied","PT5","EV2","CAT1");

select \* from Comment;

drop table Comment;

Output :



## **6.5.Tag**

Query :

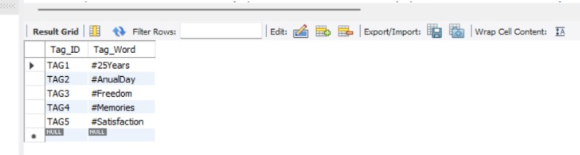
-- values in Tag --

insert into Tag values ("TAG1","#25Years"),("TAG2","#AnualDay"),("TAG3","#Freedom"),

("TAG4","#Memories"),("TAG5","#Satisfaction");

select \* from Tag;

Output :



## **6.6 Post\_has\_Tag**

Query :

-- values in Post\_Has\_Tag --

insert into Post\_Has\_Tag

values ('PT1','EV3','CAT2','TAG3'),

('PT2','EV3','CAT2','TAG5'),

('PT3','EV1','CAT1','TAG4'),

('PT4','EV2','CAT1','TAG2'),

('PT5','EV2','CAT1','TAG2'),

('PT6','EV1','CAT1','TAG1'),

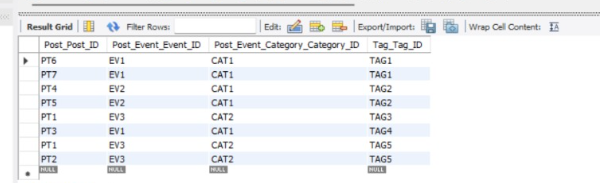
('PT7','EV1','CAT1','TAG1'),

('PT1','EV3','CAT2','TAG5');

select \* from Post\_Has\_Tag;

drop table Post\_Has\_Tag;

Output :



# **7.Implement SQL Queries to display in**

## **7.1.Mysql Workbench(using Mysql)**

### **7.1.1.Popular blog posts**

Query :

select concat("Popular Blog Posts"," are :"), Post\_Views, Post\_ID from Post where Post\_Views = (select max(Post\_Views) from Post);

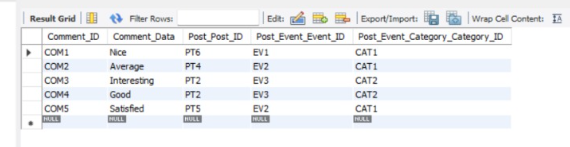
Output :



### **7.1.2.Manage comments**

Query : select \* from Comment;

Output :



### **7.1.3.Categorize posts**

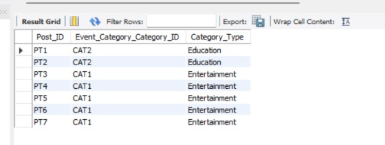
Query :

SELECT p.Post\_ID, p.Event\_Cat\_Cat\_ID, c.Category\_Type

FROM post AS p

JOIN Category AS c ON p.Event\_CatCat\_ID = c.Cat\_ID;

Output :



## **7.2.IDLE & Visual Studio Code(using Python)**

Query :

import mysql.connector as c

from tabulate import tabulate

from termcolor import colored

conn = c.connect(

user='Pbl',

password='Pbl@123',

host='localhost',

database='mydb'

)

cursor = conn.cursor()

# Define your SQL queries

sql1 = 'SELECT concat("Popular Blog Posts", " are :"), Post\_Views, Post\_ID FROM Post WHERE Post\_Views = (SELECT MAX(Post\_Views) FROM Post);'

sql2 = 'SELECT p.Post\_ID, p.Event\_Category\_Category\_ID, c.Category\_Type FROM Post AS p LEFT JOIN Category AS c ON p.Event\_Category\_Category\_ID = c.Category\_ID;'

sql3 = 'SELECT \* FROM Comment;'

# Define a function to execute and print query results as a colorful table

def execute\_and\_print\_query(cursor, query):

cursor.execute(query)

result = cursor.fetchall()

print(colored("Query: ", "blue"), colored(query, "blue"))

headers = [colored(i[0], "green") for i in cursor.description]

colored\_results = [[colored(str(cell), "yellow") for cell in row] for row in result]

print(tabulate(colored\_results, headers=headers, tablefmt="fancy\_grid"))

print()

# Execute and print results for each query

execute\_and\_print\_query(cursor, sql1)

execute\_and\_print\_query(cursor, sql2)

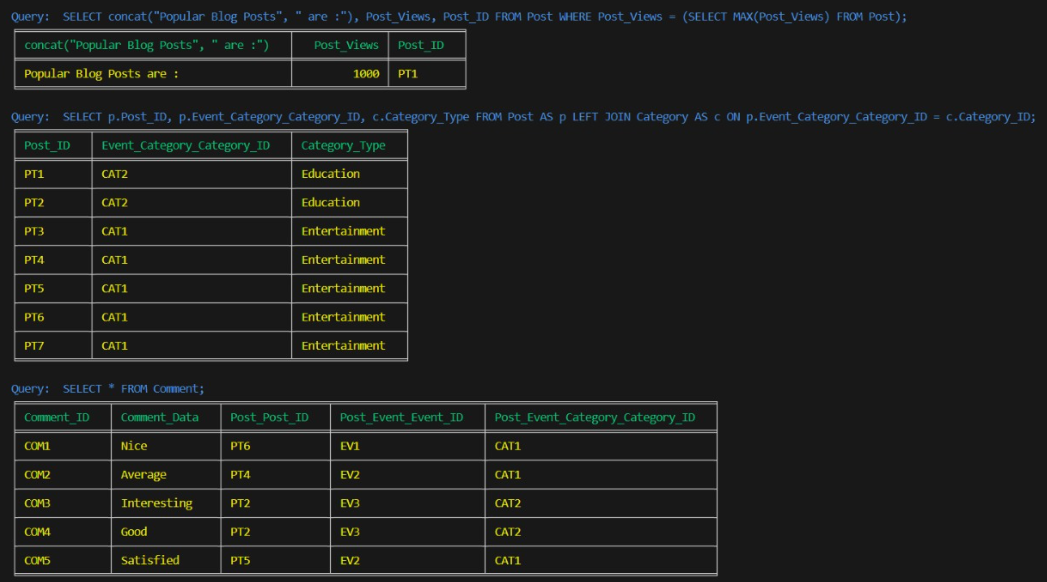
execute\_and\_print\_query(cursor, sql3)

# Close the cursor and connection

cursor.close()

conn.close()

Output :



# **8.Remarks**