**Types of Access in MySQL**

When accessing a MySQL server **remotely**, the method depends on how the user is defined:

| **Type of Access** | **Example User Definition** | **Meaning** |
| --- | --- | --- |
| Local Access | 'john'@'localhost' | Can connect **only from same machine** |
| Remote IP Access | 'john'@'192.168.1.100' | Can connect only from **this IP address** |
| Wildcard IP Access | 'john'@'%' | Can connect from **any host** |
| Hostname Access | 'john'@'host.example.com' | Can connect only from **specific hostname** |

Tip: '%' (wildcard) is commonly used for testing or remote access, but be cautious with it in production for security reasons.

Create Multiple Users

--Local user (from same server)

CREATE USER 'user\_local'@'localhost' IDENTIFIED BY 'local123';

-- Remote user from any IP

CREATE USER 'user\_remote'@'%' IDENTIFIED BY 'remote123';

-- Specific IP (replace with your actual IP if needed)

CREATE USER 'user\_ip'@'192.168.1.9' IDENTIFIED BY 'ipuser123';

**GRANT Permissions**

**Example 1: Give read access (SELECT) on company\_db to user\_remote**

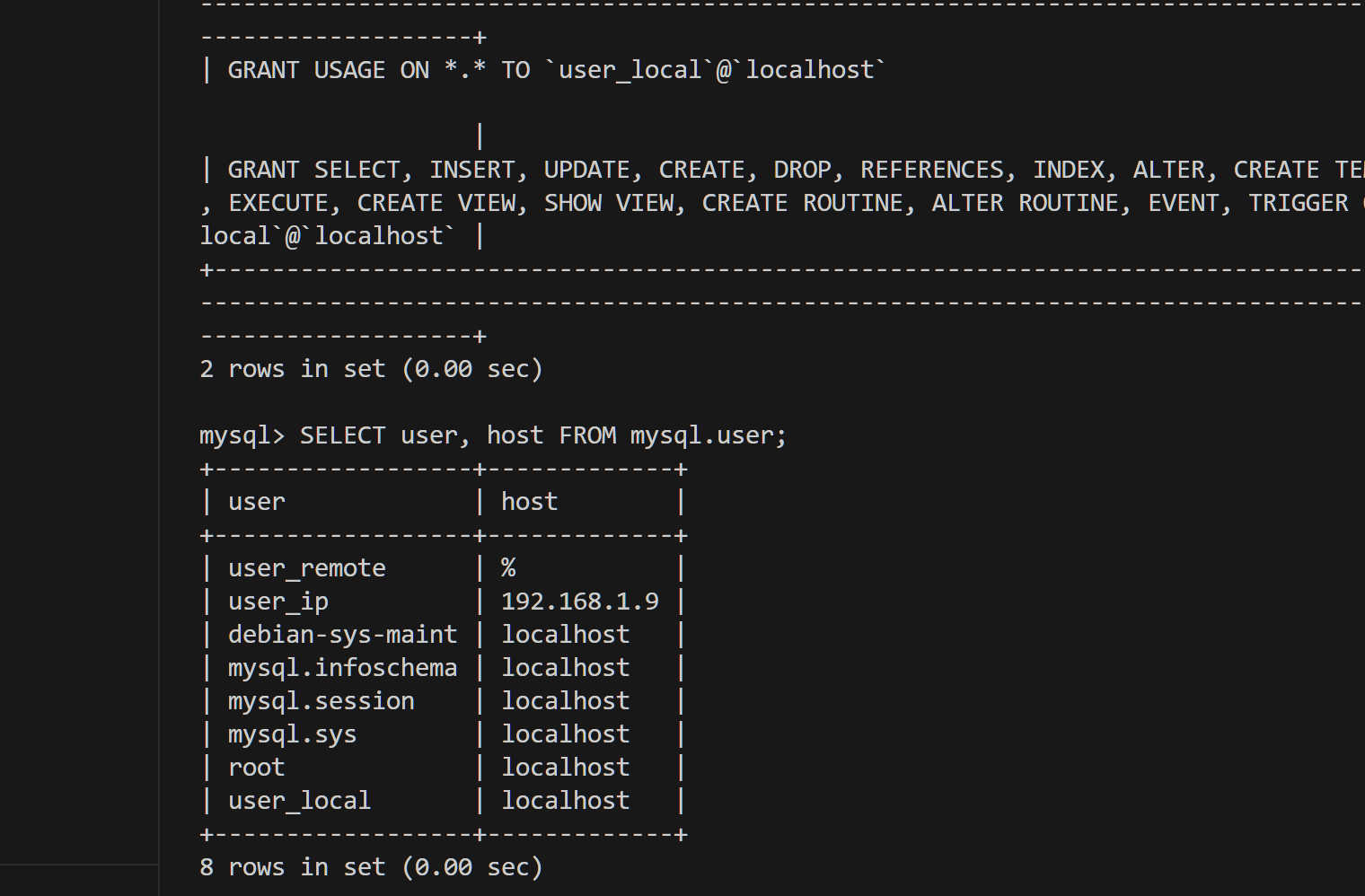
GRANT SELECT ON company\_db.\* TO 'user\_remote'@'%';

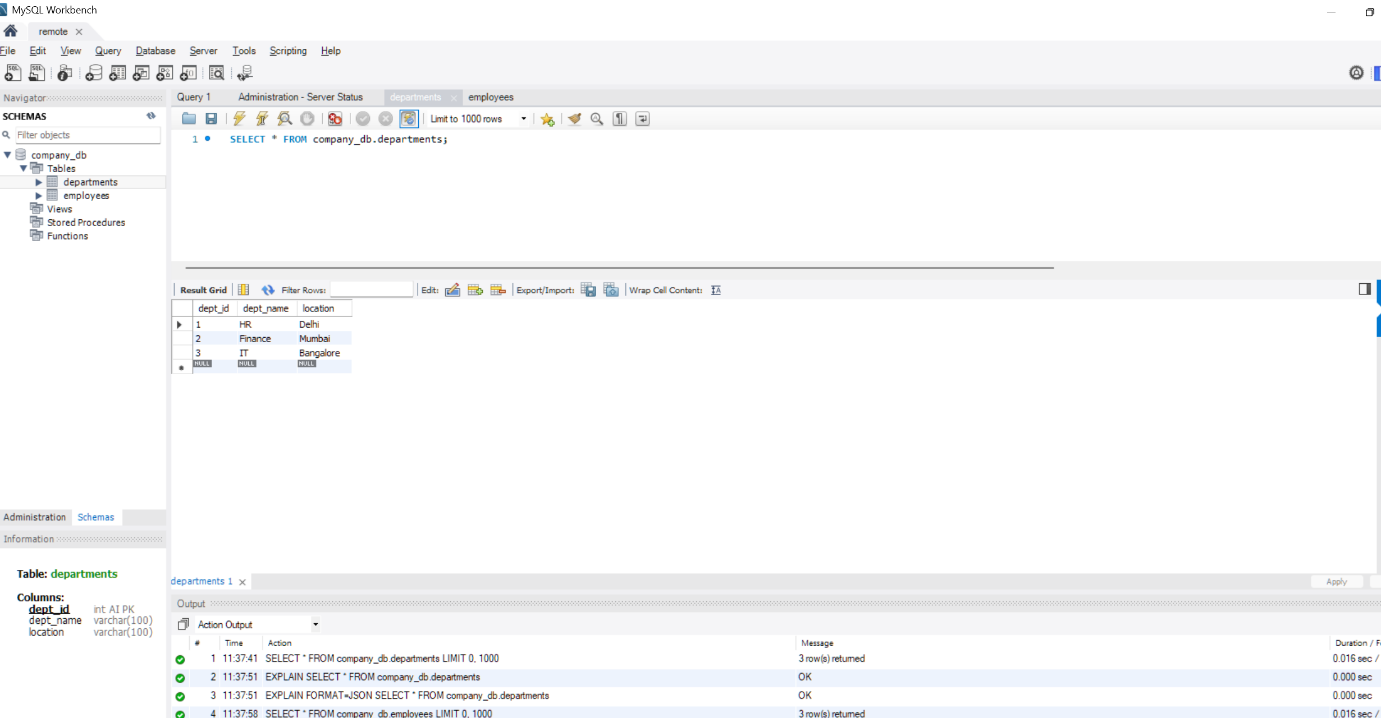
**Example 2: Give full access to only employees table for user\_ip**

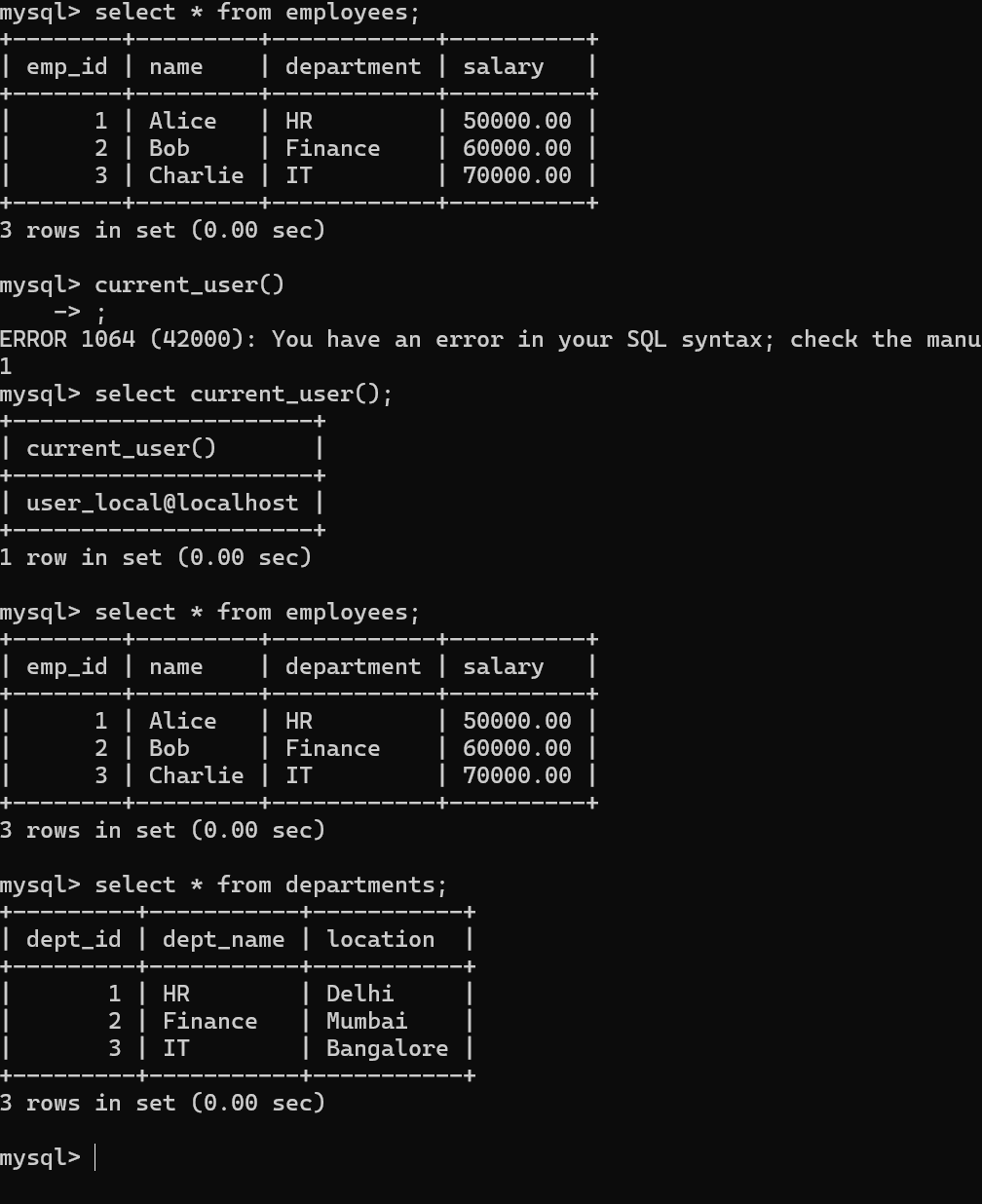
GRANT SELECT, INSERT, UPDATE, DELETE ON company\_db.employees TO 'user\_ip'@'192.168.1.9';

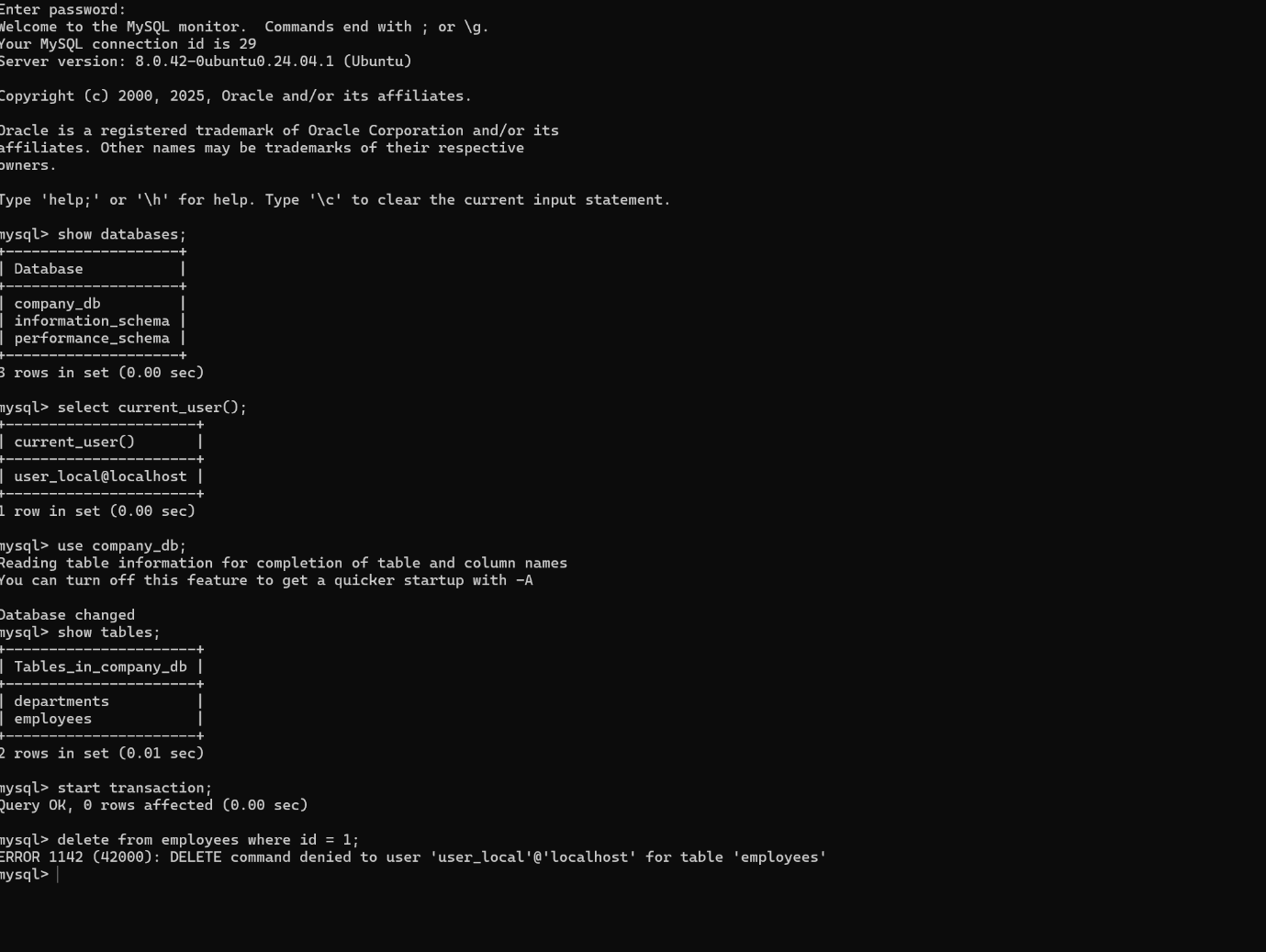
**Example 3: Give ALL privileges on the database to user\_local**

GRANT ALL PRIVILEGES ON company\_db.\* TO 'user\_local'@'localhost';









--Revoke DELETE Privilege Only

REVOKE DELETE ON company\_db.\* FROM 'user\_local'@'localhost';

flush privileges;

SHOW GRANTS FOR 'user\_local'@'localhost';

--Log in as user\_local from ubuntu server

-- mysql -u user\_local -p;

--Switch to company\_db database

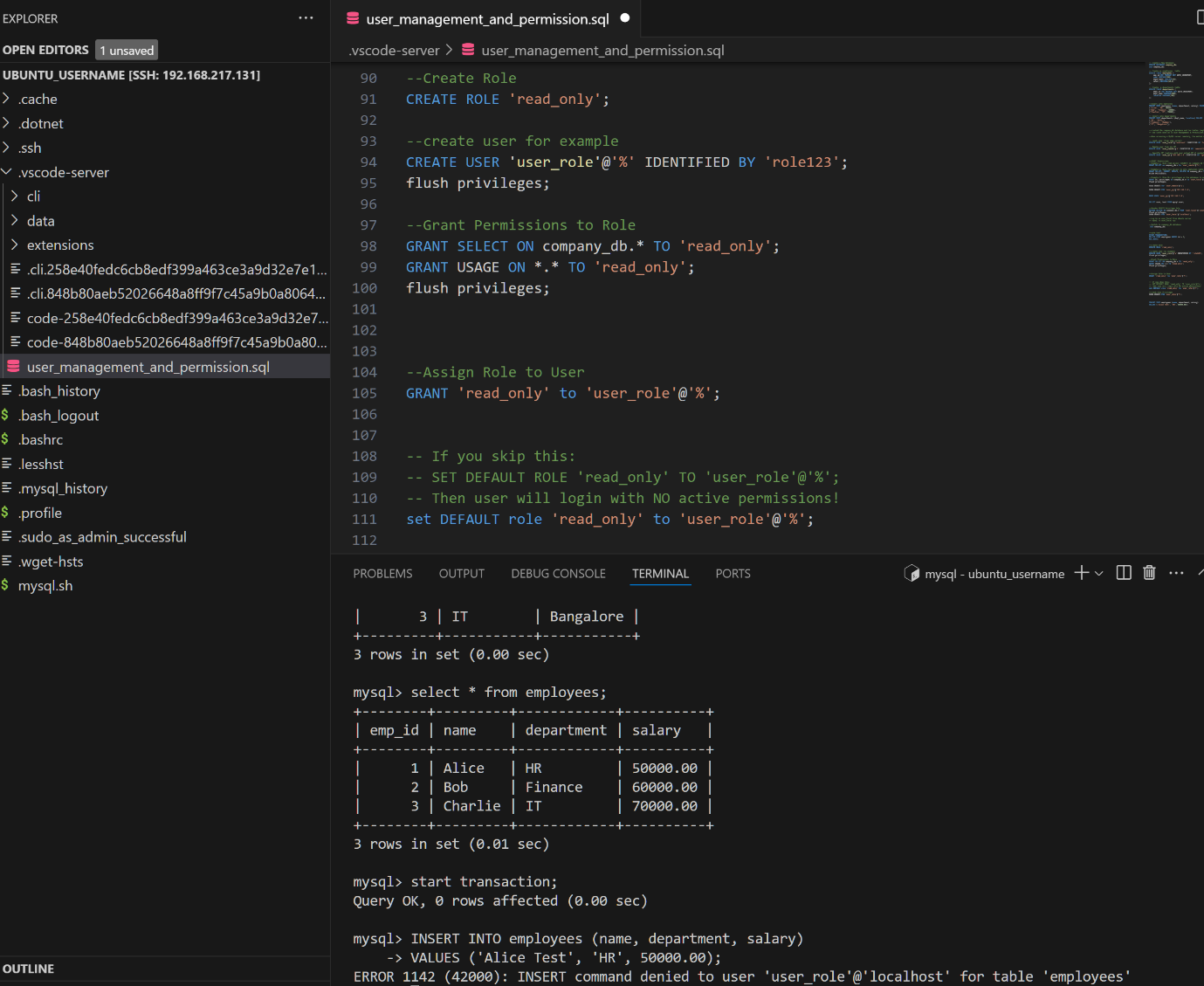
 USE company\_db;

--safe test

START TRANSACTION;

DELETE FROM employees WHERE id = 1;

ROLLBACK;



--Create Role

CREATE ROLE 'read\_only';

--create user for example

CREATE USER 'user\_role'@'%' IDENTIFIED BY 'role123';

flush privileges;

--Grant Permissions to Role

GRANT SELECT ON company\_db.\* TO 'read\_only';

GRANT USAGE ON \*.\* TO 'read\_only';

flush privileges;

--Assign Role to User

GRANT 'read\_only' to 'user\_role'@'%';

-- If you skip this:

-- SET DEFAULT ROLE 'read\_only' TO 'user\_role'@'%';

-- Then user will login with NO active permissions!

set DEFAULT role 'read\_only' to 'user\_role'@'%';

--Check User Privileges

SHOW GRANTS FOR 'user\_role'@'%';

--safe test for user\_role, trying to insert row into table employees,

--but it do not have permission to do so.

start TRANSACTION;

INSERT INTO employees (name, department, salary)

VALUES ('Alice Test', 'HR', 50000.00);

Basic Configuration & Secure MySQL means setting up MySQL in a safe and controlled way to protect your data. Here’s how:

* Change the MySQL configuration file to allow or block remote access using something called bind-address. Setting it to 127.0.0.1 allows only local access. Setting it to 0.0.0.0 allows access from anywhere.
* Run the mysql\_secure\_installation command. This is a built-in script that helps you secure your MySQL server. It asks you questions like setting a root password, removing anonymous users, removing test databases, and disallowing remote root login.
* Disable remote login for the root user so no one can connect to MySQL as root from another computer. This reduces the risk of being hacked.
* Remove any unknown or unnecessary users or databases that were created by default.
* Always set a strong password for every user, especially the root user.
* Keep regular backups and updates for MySQL to avoid security issues.

**To download sample database employees**

# Step 1: Download

wget https://github.com/datacharmer/test\_db/archive/refs/heads/master.zip

# Step 2: Unzip

sudo apt install unzip -y

unzip master.zip

cd test\_db-master

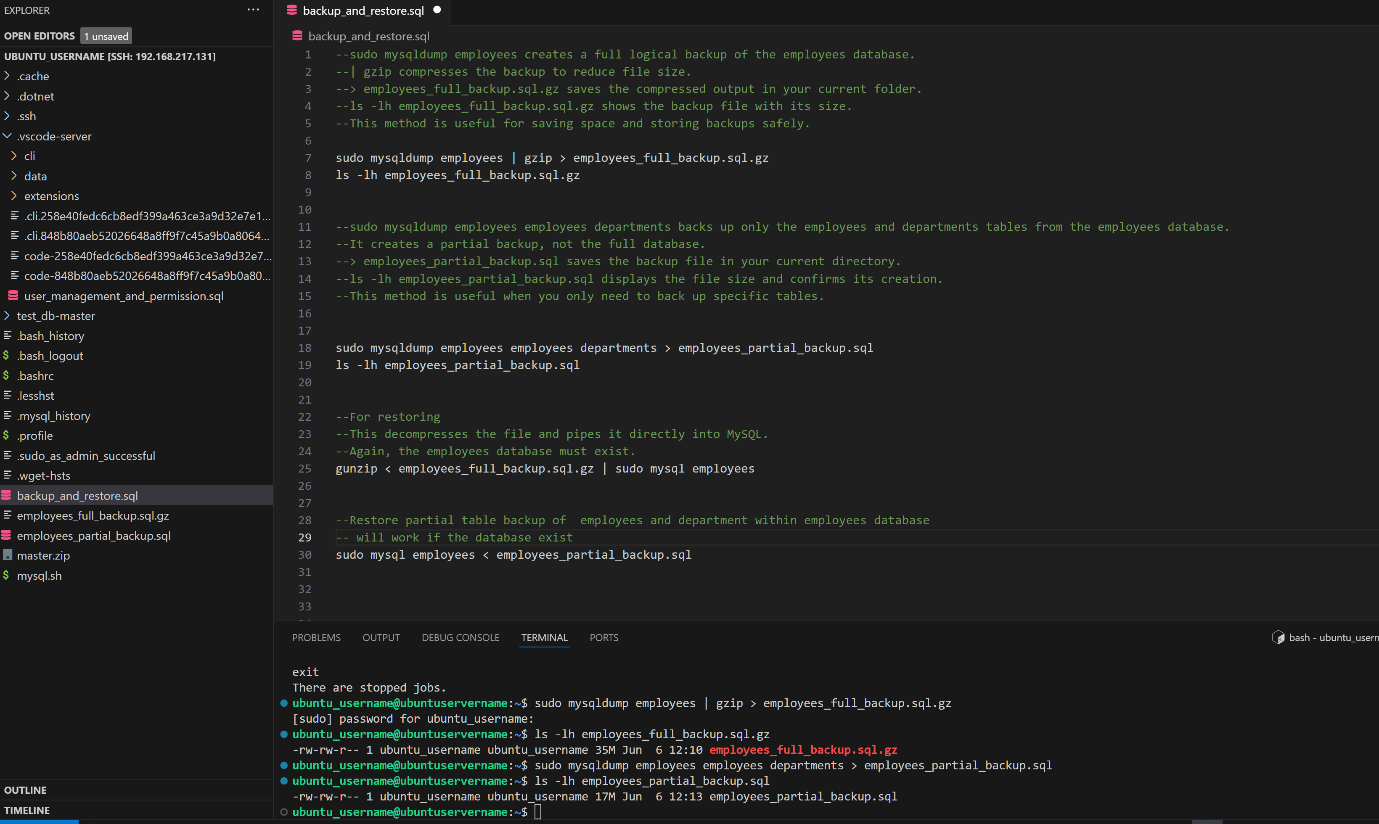
# Run the entire dataset loader

mysql -u root -p < employees.sql

**Logical Backup using MySQL dump** with examples from the **employees** database:

**What is Logical Backup?**

* It means saving the **data and structure** of your database into a text file with SQL commands.
* This file can be used later to **restore** the database by running those commands.



**What is mysqldump?**

* A tool that comes with MySQL to create these logical backups.
* It exports your database as SQL statements (like CREATE TABLE, INSERT INTO).

**Backups with mysqldump**

**Full Backup:**

You back up the **entire database**.

Example: Back up the whole employee’s database.

sudo mysqldump employees > employees\_full\_backup.sql

**Partial Backup:**

You back up **only some tables** or parts of the database.

Example: Back up only the employees and departments tables from the

employee’s database

sudo mysqldump employees employees departments >

employees\_partial\_backup.sql

**Compress Backups:**

Backups can be very big, so you can compress them on the fly.

Example: Compress full backup using gzip to save space.

sudo mysqldump employees | gzip > employees\_full\_backup.sql.gz

**Full Backup of employees using XtraBackup**

**1. Install Percona XtraBackup**

sudo apt update

sudo apt install percona-xtrabackup-80

2. **Take a Full Physical Backup**

sudo xtrabackup --backup --datadir=/var/lib/mysql --target-dir=/backup/employees\_full

--datadir=/var/lib/mysql → where your MySQL data is stored (default location).

--target-dir=/backup/employees\_full → where to save the backup.

This creates a physical backup of **all databases**, including employees.

3.  **Prepare the Backup**

Before restoring, run this to apply logs and make the backup ready:

sudo xtrabackup --prepare --target-dir=/backup/employees\_full

4. **Restore the Backup**

Stop Mysql first

sudo systemctl stop mysql

Now restore the data:

sudo rm -rf /var/lib/mysql/\*

sudo xtrabackup --copy-back --target-dir=/backup/employees\_full

**Set correct permissions**:

sudo chown -R mysql: mysql /var/lib/mysql

* This gives permission to the mysql user so MySQL can **read and write** its own data.
* Without this, MySQL **won’t start** or will throw "permission denied" errors.

**Start MySQL again:**

sudo systemctl start mysql

What You Just Did (with employees):

* Took a physical backup of **employees** (and all other databases).
* Prepared the backup for restore.
* Restored it safely into MySQL data directory.
* Your employee’s database is now restored exactly as it was — including data, tables, indexes.

**Why do you need to prepare the backup before restoring?**

**Reason:**

* When XtraBackup takes a backup, some data may still be "in memory" or mid-write (like a half-written transaction).
* The prepare step applies the transaction logs (like a journal) to ensure the data is consistent and safe to restore.

**Think of it like:**

Taking a photo of a moving object — you need to process it before it's clear.

sudo xtrabackup --prepare --target-dir=/backup/employees\_full

This "cleans up" and finalizes the backup, so it's safe to restore.

**MySQL Replication and GTID Setup Guide (On Ubuntu Server in VMware)**

**PART 1: Master-Slave Replication using Binary Log**

**Step 1: Prepare Master MySQL Instance (port 3306)**

1. Edit the MySQL config file:

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

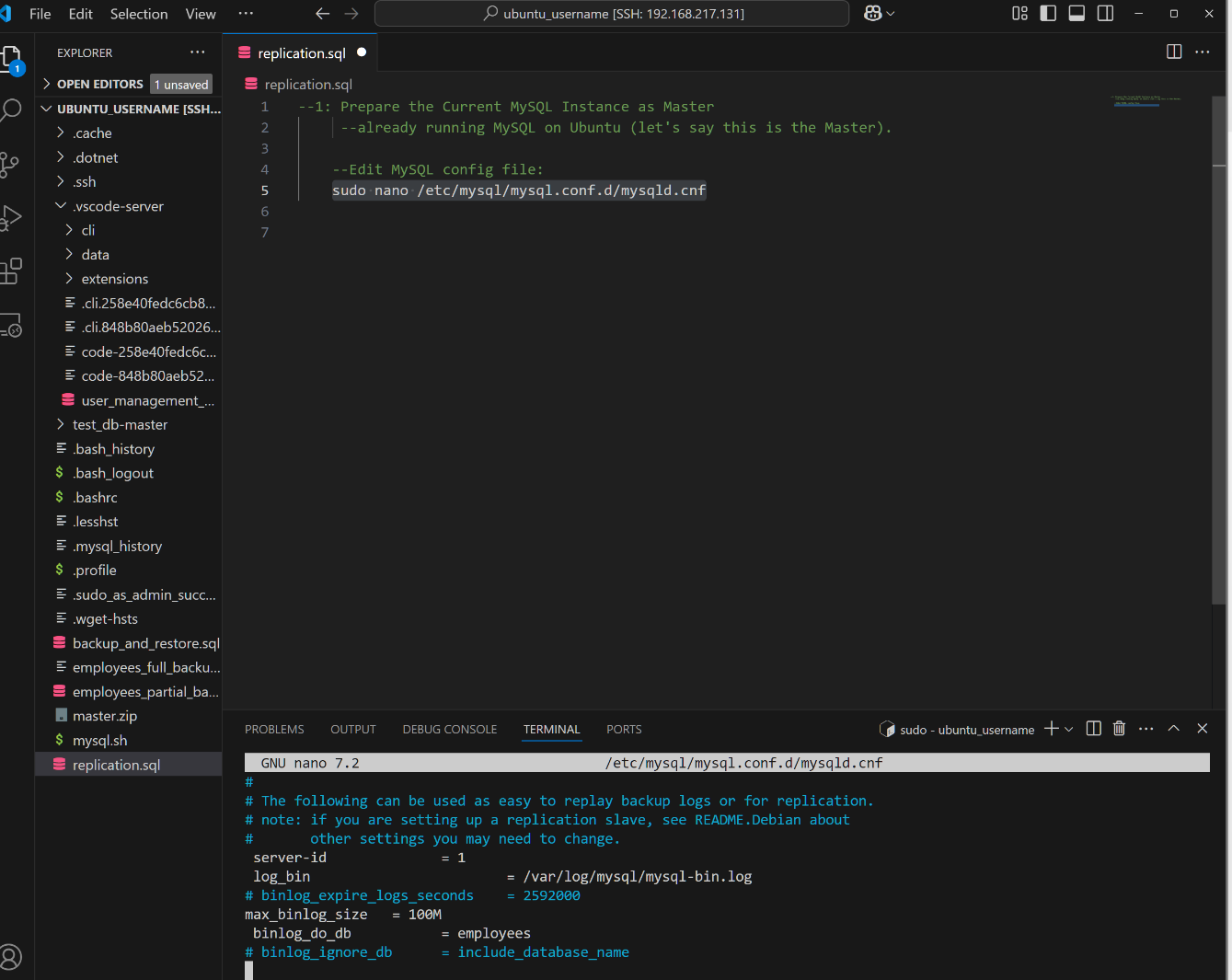
Add or edit the following lines:

server-id = 1

log\_bin = /var/log/mysql/mysql-bin.log

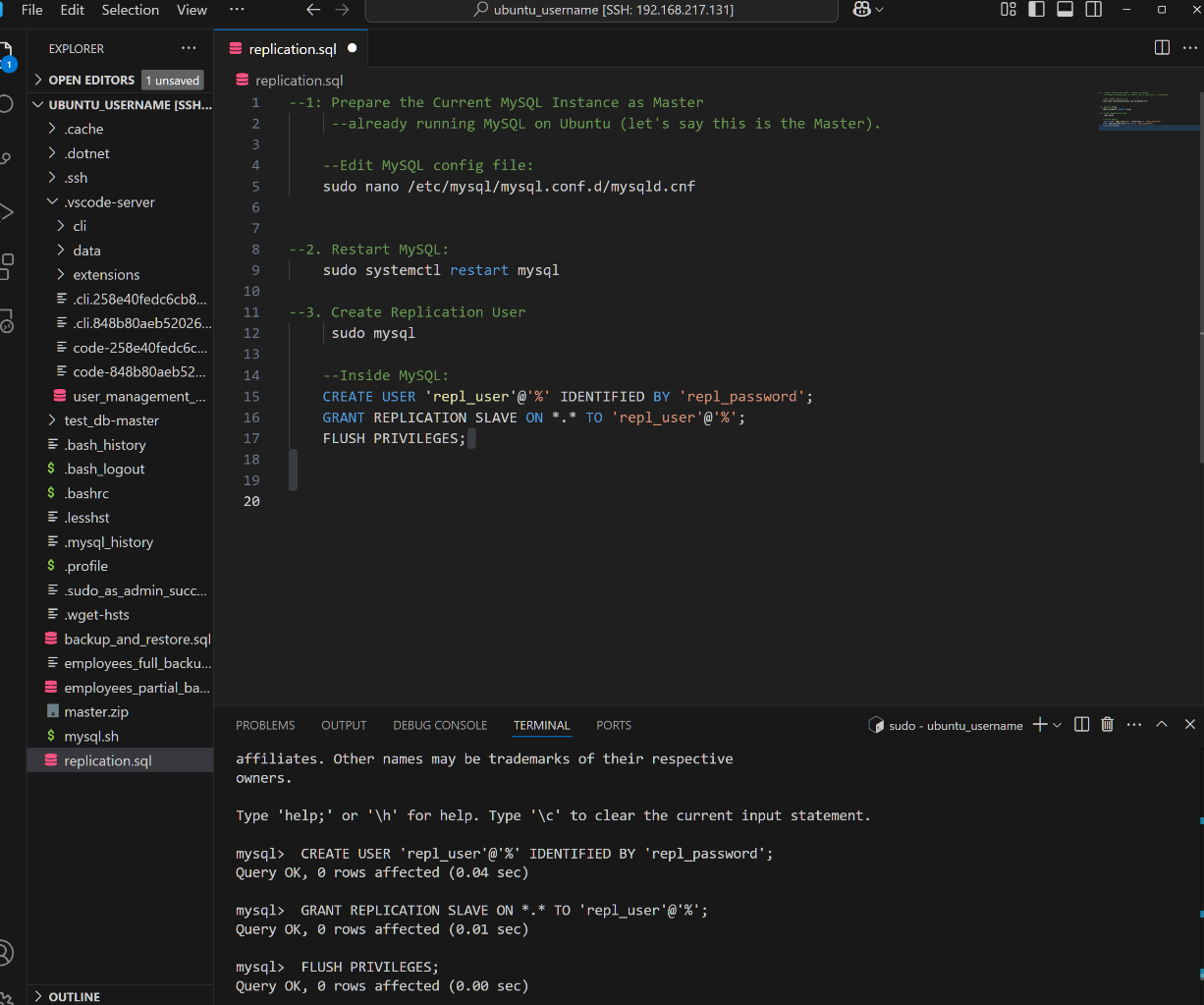
binlog\_do\_db = employees # Optional: replicate only this DB

bind-address = 0.0.0.0 # Allow external access



1. Restart MySQL:

sudo systemctl restart mysql

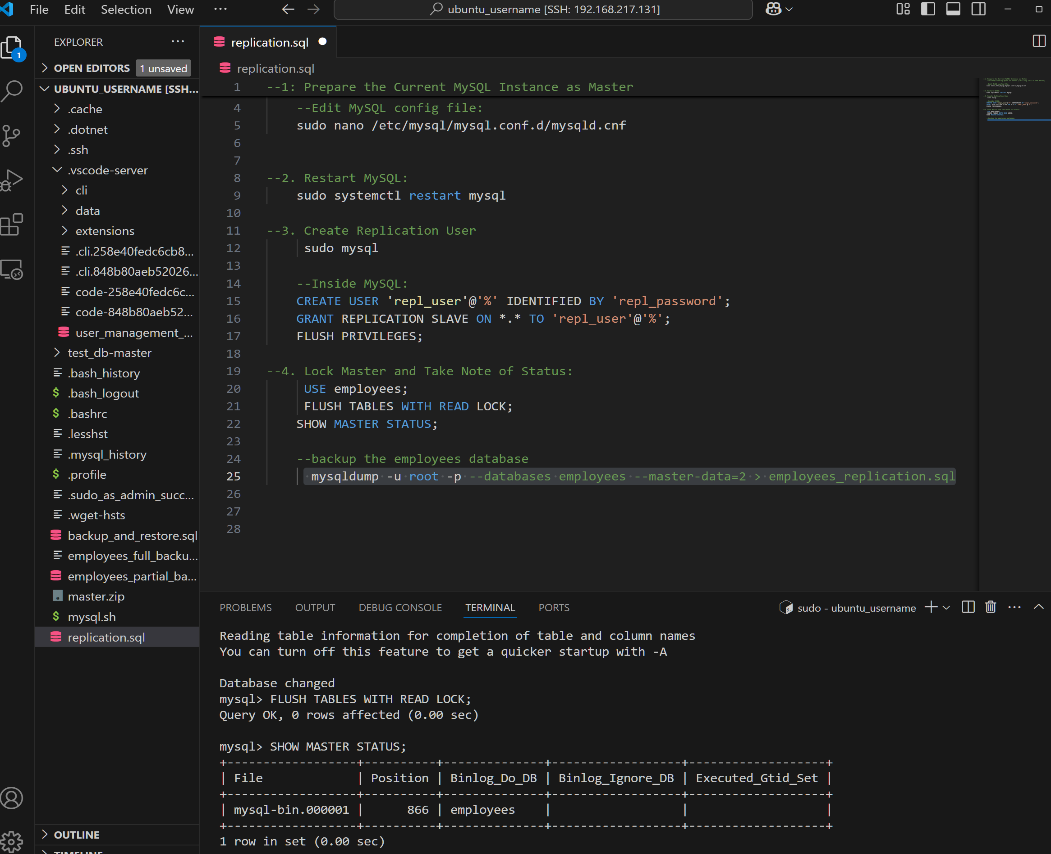


1. Create Replication User:

CREATE USER 'repl\_user'@'%' IDENTIFIED WITH mysql\_native\_password BY 'repl\_password';

GRANT REPLICATION SLAVE ON \*. \* TO 'repl\_user'@'%';

FLUSH PRIVILEGES;



1. Lock and Export the Database:

USE employees;

FLUSH TABLES WITH READ LOCK;

SHOW MASTER STATUS;

In a new terminal:

mysqldump -u root -p --databases employees --master-data=2 > employees\_replication.sql

Then unlock the tables from original session:

UNLOCK TABLES;

**Step 2: Setup MySQL Slave Instance (port 3307) on Same Ubuntu Server**

1. Create directories:

sudo mkdir /etc/mysql-slave

sudo mkdir /var/lib/mysql-slave

sudo mkdir /var/log/mysql-slave

sudo chown mysql:mysql /var/log/mysql-slave

1. Copy and update config:

sudo cp -r /etc/mysql/mysql.conf.d /etc/mysql-slave/

sudo cp /etc/mysql/my.cnf /etc/mysql-slave/

Edit /etc/mysql-slave/my.cnf:

!includedir /etc/mysql-slave/mysql.conf.d/

Edit /etc/mysql-slave/mysql.conf.d/mysqld.cnf:

[mysqld]

port = 3307

datadir = /var/lib/mysql-slave

socket = /var/run/mysqld/mysqld\_slave.sock

pid-file = /var/run/mysqld/mysqld\_slave.pid

log-error = /var/log/mysql-slave/error.log

server-id = 2

relay\_log = /var/log/mysql-slave/mysql-relay-bin

skip-networking = 0

bind-address = 0.0.0.0

1. Initialize Data Directory:

sudo mysqld --initialize-insecure --datadir=/var/lib/mysql-slave --user=mysql

1. Create New Systemd Service:

sudo cp /lib/systemd/system/mysql.service /etc/systemd/system/mysql-slave.service

Edit:

sudo nano /etc/systemd/system/mysql-slave.service

# Change ExecStart to:

ExecStart=/usr/sbin/mysqld --defaults-file=/etc/mysql-slave/my.cnf

Reload and start:

sudo systemctl daemon-reexec

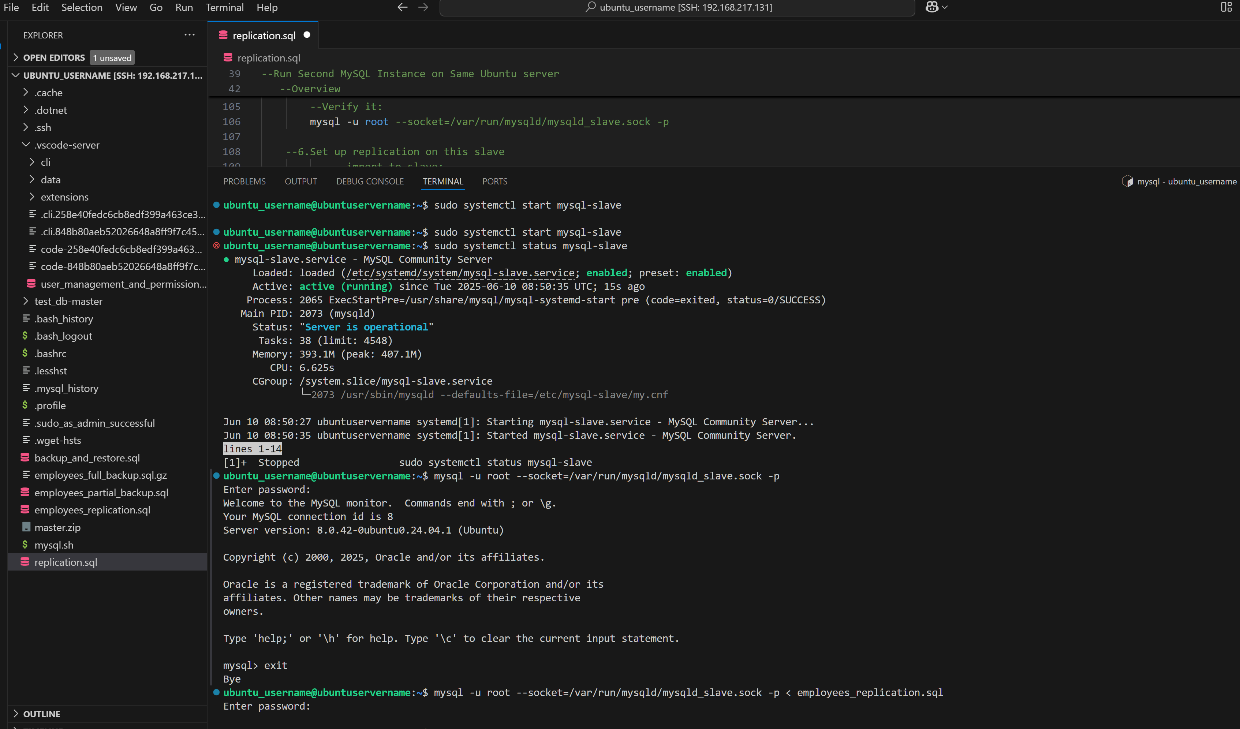
sudo systemctl daemon-reload

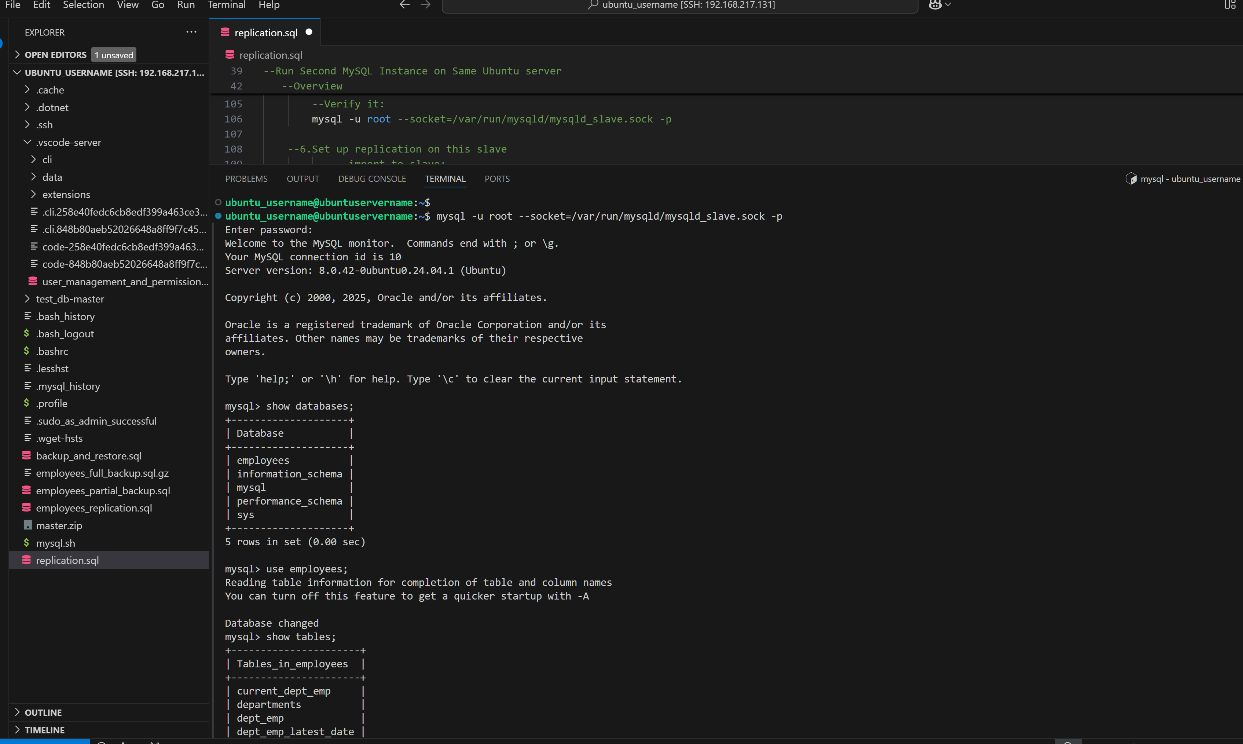
sudo systemctl start mysql-slave

sudo systemctl enable mysql-slave

1. Configure Slave:

mysql -u root --socket=/var/run/mysqld/mysqld\_slave.sock -p





Import the dump:

mysql -u root --socket=/var/run/mysqld/mysqld\_slave.sock -p < employees\_replication.sql

Set replication:

CHANGE MASTER TO

MASTER\_HOST='192.168.217.131',

MASTER\_PORT=3306,

MASTER\_USER='repl\_user',

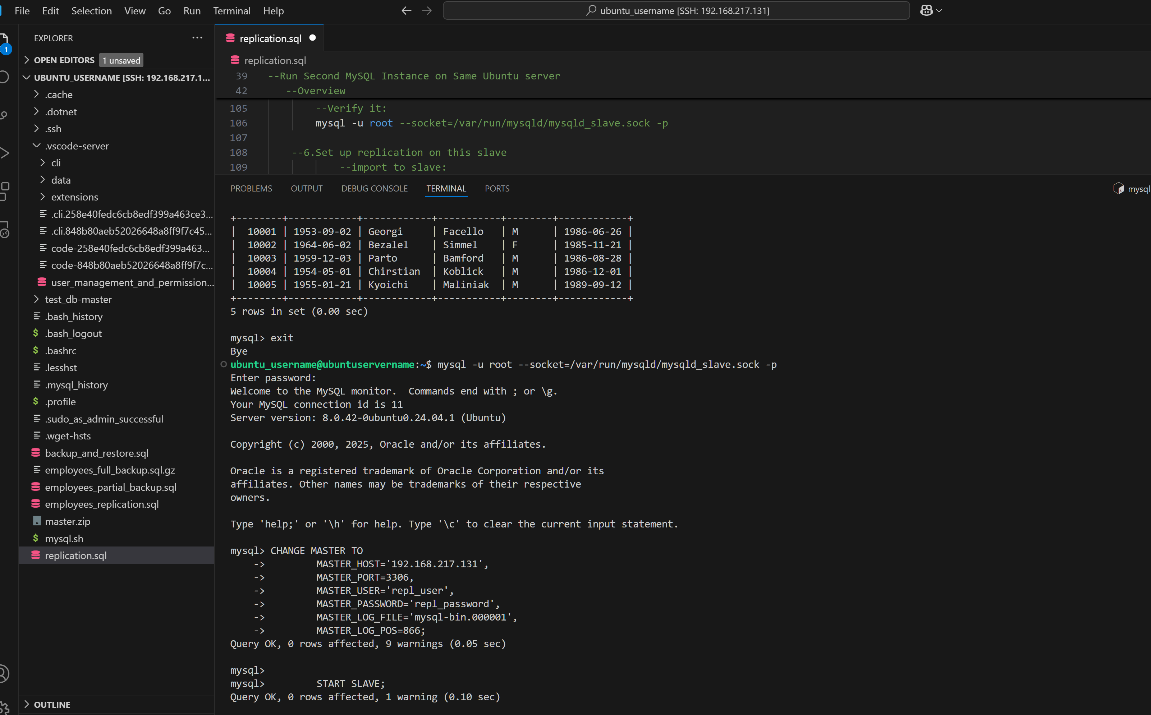
MASTER\_PASSWORD='repl\_password',

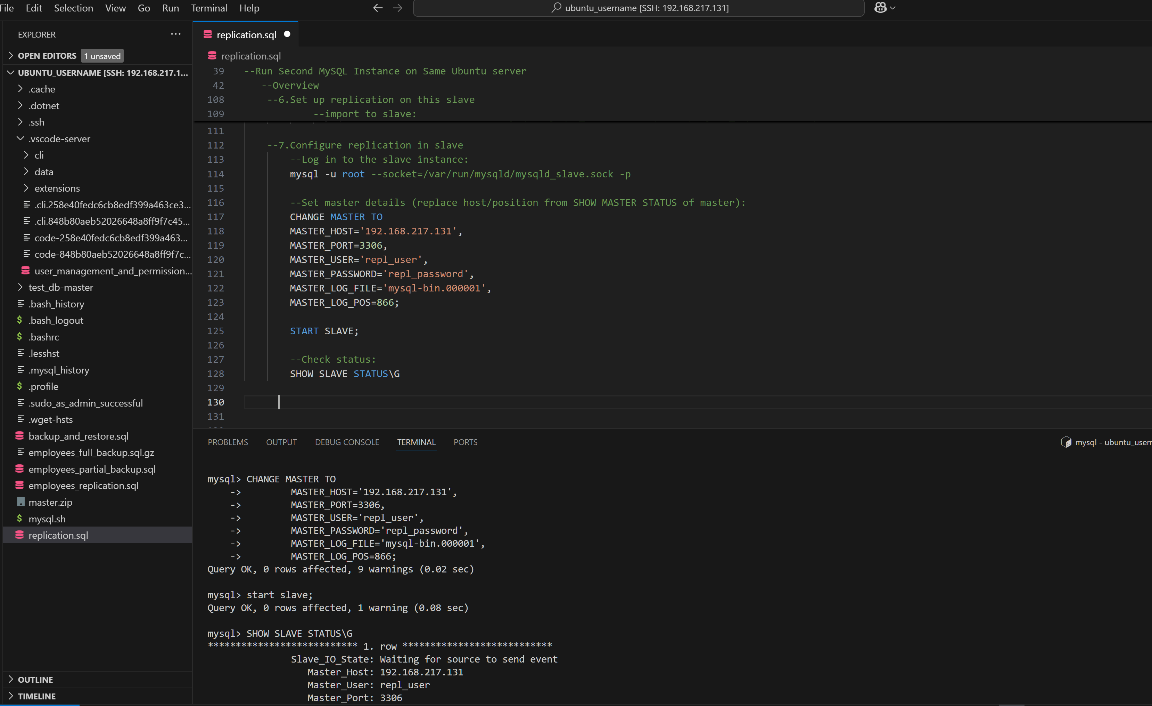
MASTER\_LOG\_FILE='mysql-bin.000001',

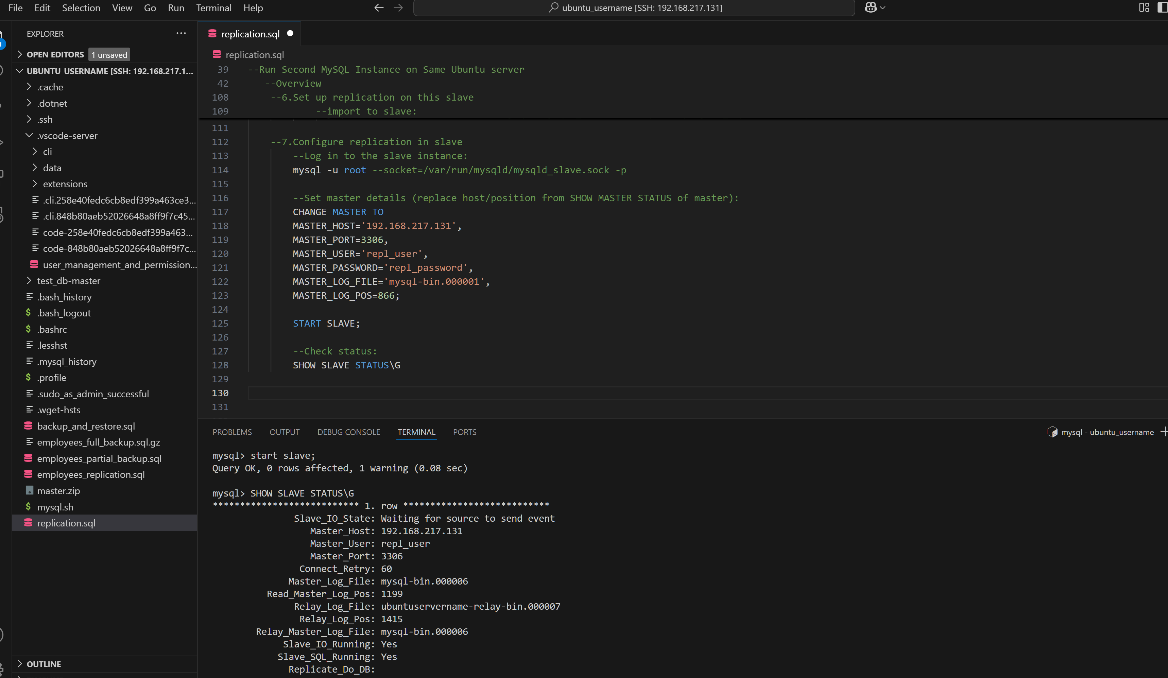
MASTER\_LOG\_POS=866;

START SLAVE;

SHOW SLAVE STATUS\G





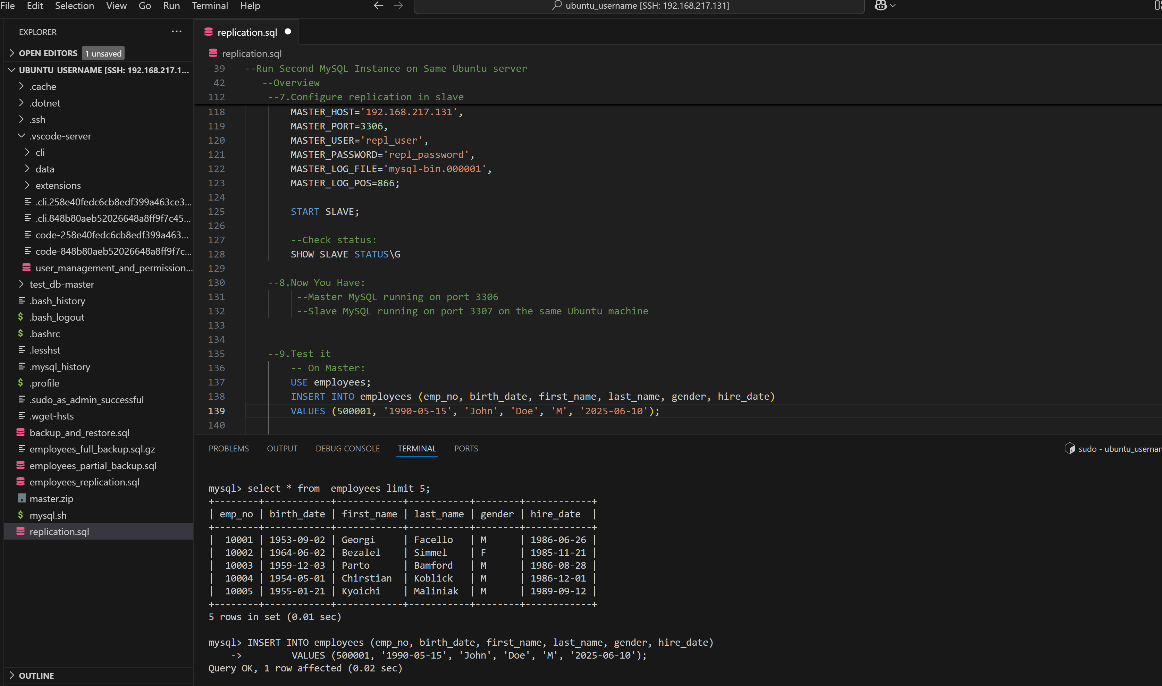


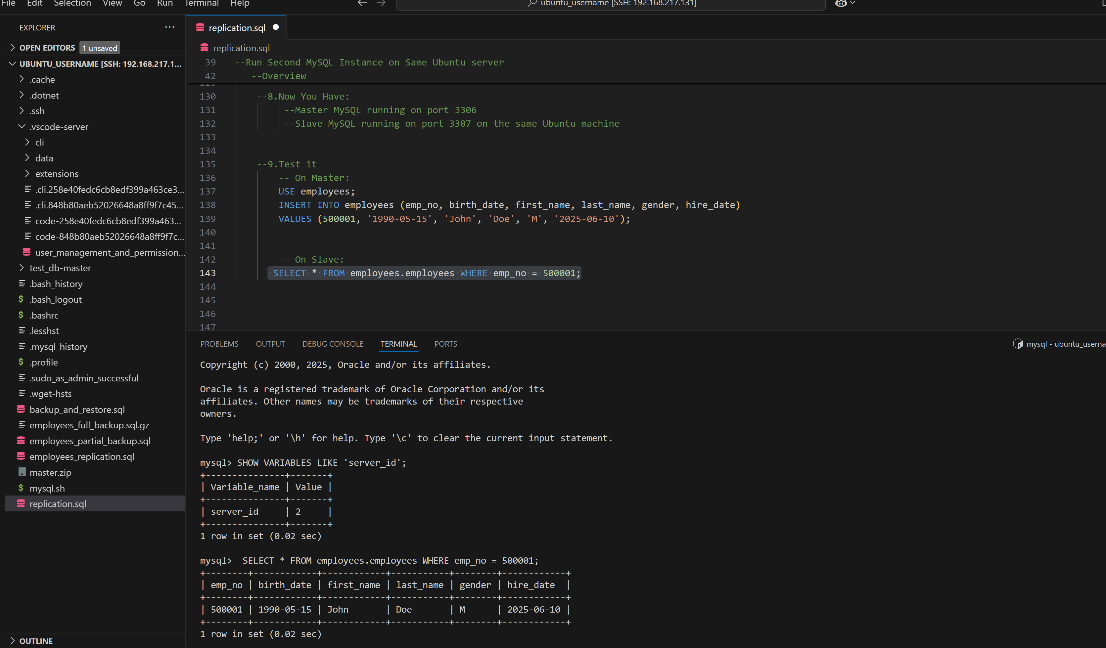
1. Test: Insert into Master:

USE employees;

INSERT INTO employees (emp\_no, birth\_date, first\_name, last\_name, gender, hire\_date)

VALUES (500001, '1990-05-15', 'John', 'Doe', 'M', '2025-06-10');





Check Slave:

SELECT \* FROM employees.employees WHERE emp\_no = 500001;

**PART 2: GTID-Based Replication**

**Goal: Convert Existing Binary Log-Based Replication to GTID-based.**

Step 1: Update Master (3306)

EDIT config:

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

Add:

enforce\_gtid\_consistency = ON

gtid\_mode = ON

log\_slave\_updates = ON

binlog\_format = ROW

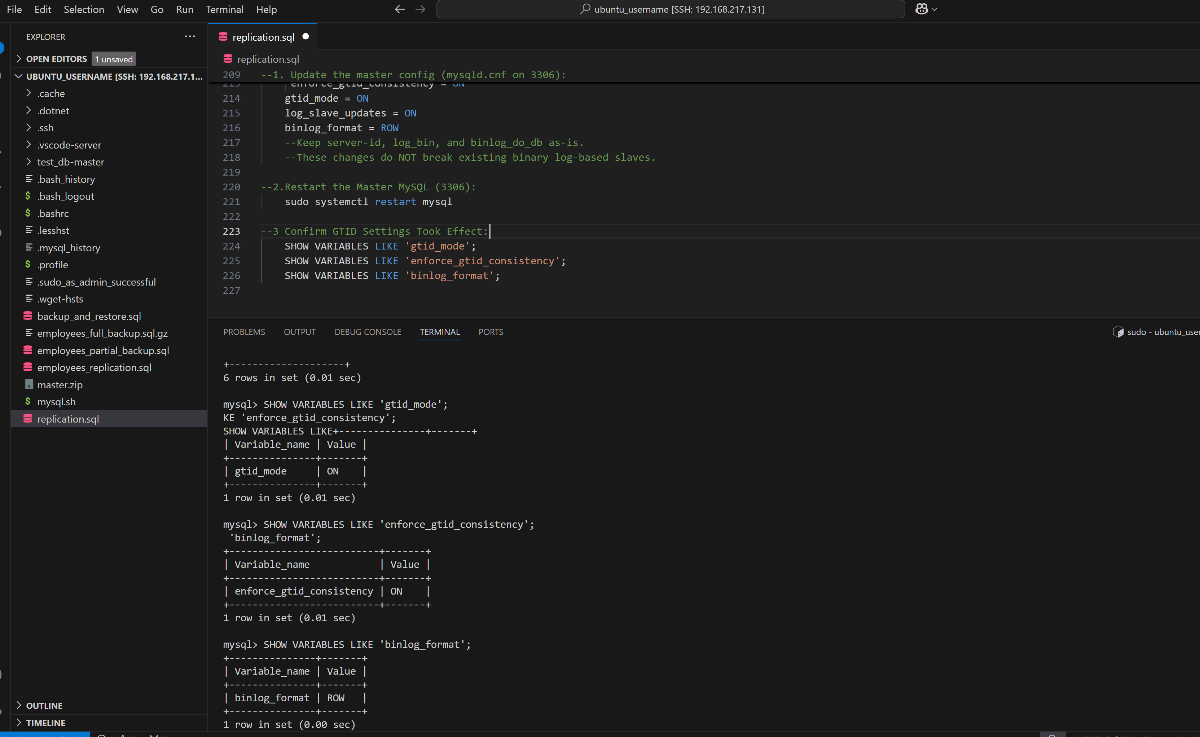
Restart:

sudo systemctl restart mysql

Check:

SHOW VARIABLES LIKE 'gtid\_mode';

SHOW VARIABLES LIKE 'enforce\_gtid\_consistency';



Step 2: Update Slave (3307)

Stop Slave

STOP SLAVE;

Edit slave config:

sudo nano /etc/mysql-slave/mysql.conf.d/mysqld.cnf

Add/update:

server-id = 2

gtid\_mode = ON

enforce\_gtid\_consistency = ON

log\_slave\_updates = ON

log\_bin = /var/log/mysql/mysql-bin.log

binlog\_format = ROW

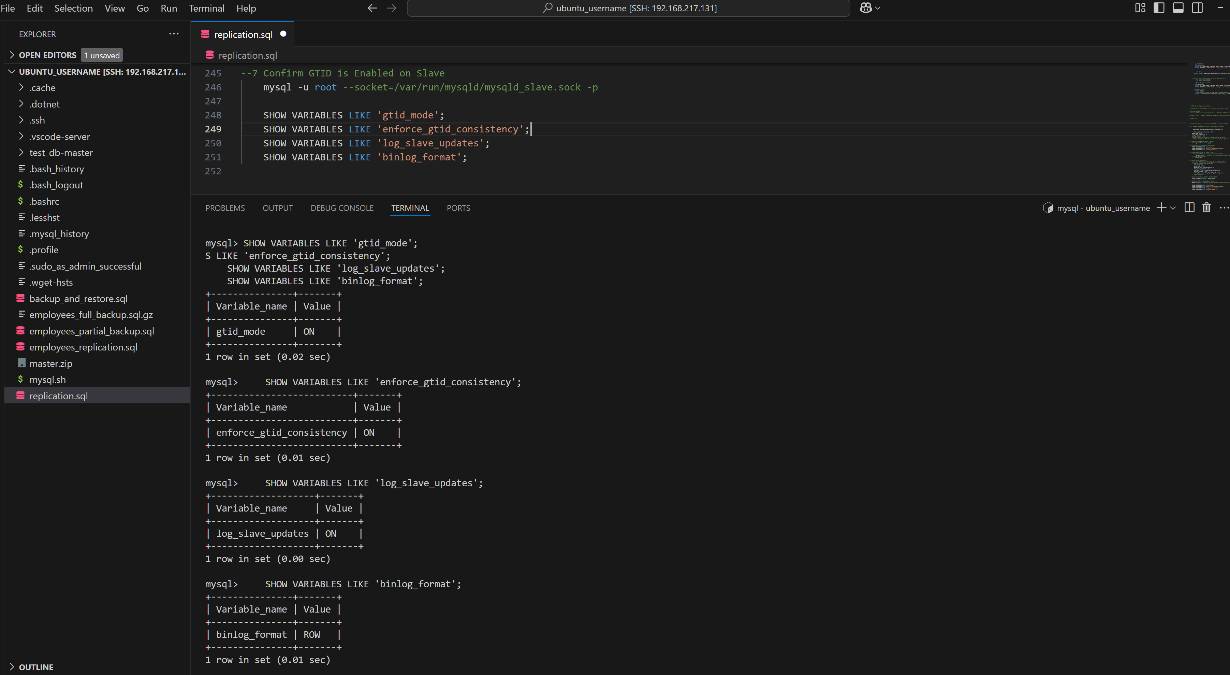
relay\_log = /var/log/mysql/mysql-relay-bin

Restart slave:

sudo systemctl restart mysql-slave

Confirm:

SHOW VARIABLES LIKE 'gtid\_mode';



Step 3: Reconfigure GTID-Based Replication

Reset and reconfigure:

RESET SLAVE ALL;

CHANGE MASTER TO

MASTER\_HOST='192.168.217.131',

MASTER\_USER='repl\_user',

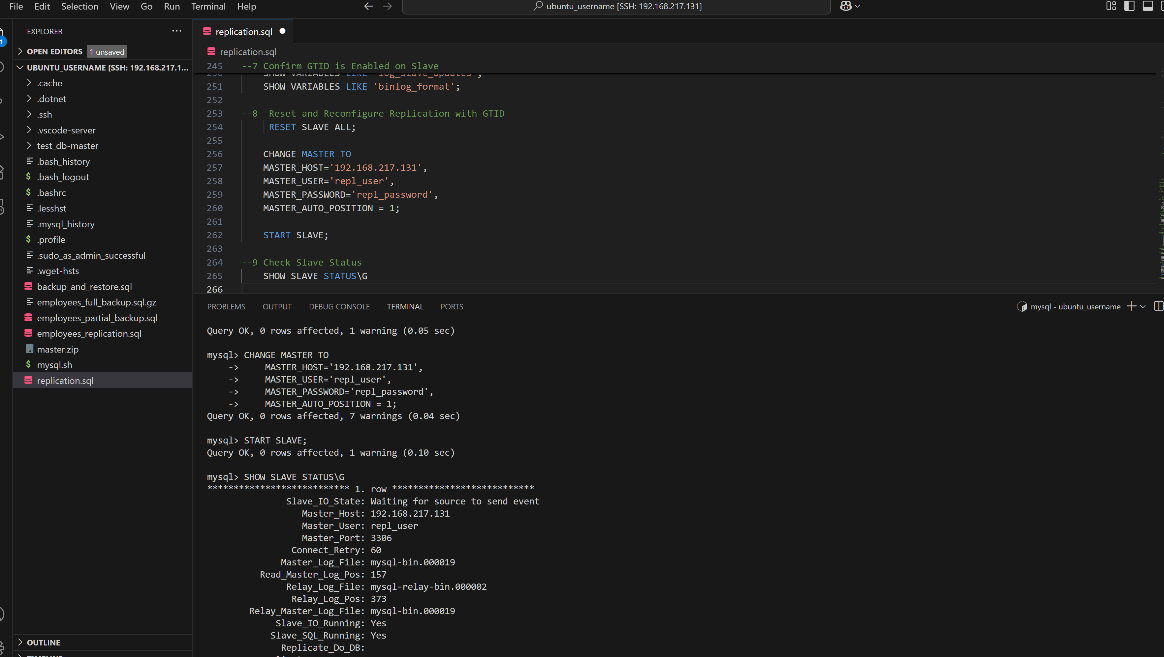
MASTER\_PASSWORD='repl\_password',

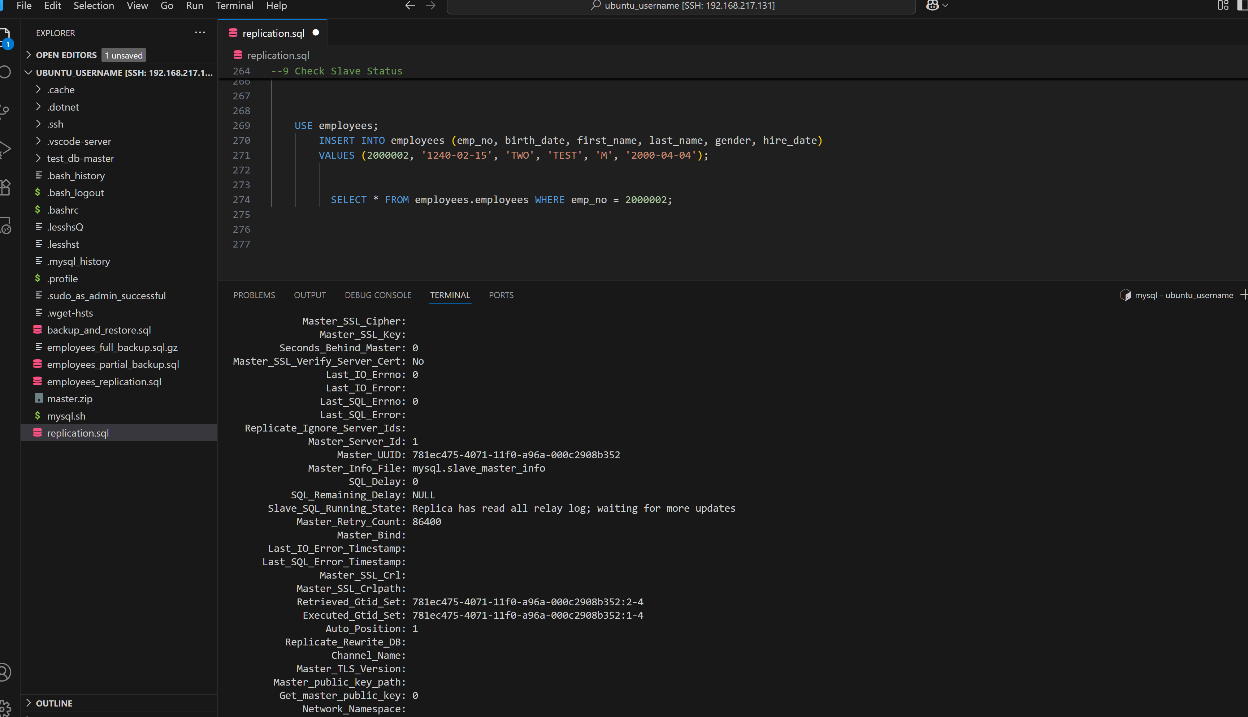
MASTER\_AUTO\_POSITION = 1;

START SLAVE;

Check:

SHOW SLAVE STATUS\G:





**Replication Failure Scenarios**

**Goal: Learn how to simulate, detect, fix replication issues.**

**1. Break the Replication (Intentionally)**

**: Insert duplicate key on master**

**-- On Master:**

USE employees;

        INSERT INTO employees (emp\_no, birth\_date, first\_name, last\_name, gender, hire\_date)

        VALUES (6000001, '1890-05-15', 'Ray', 'Doe', 'F', '2025-04-10');

❗ On slave, this will throw **Error 1062: Duplicate entry**

Run:

SHOW SLAVE STATUS\G

Check:

Slave\_SQL\_Running: No

Last\_SQL\_Error shows the duplicate key issue

**2.Fix Replication (Skip the error)**

STOP SLAVE;

SET GLOBAL SQL\_SLAVE\_SKIP\_COUNTER = 1;

START SLAVE;

SHOW SLAVE STATUS\G

**3. Resync the Slave from Master (Manual Resync)**

**If your slave is completely out of sync:**

**Stop the slave:**

STOP SLAVE;

**Re-dump from master:**

mysqldump -u root -p --databases employees --master-data=2 > fresh.sql

**Import into slave:**

mysql -u root --socket=/path/to/slave.sock -p < fresh.sql

**Reset slave and reconfigure:**

RESET SLAVE ALL;

CHANGE MASTER TO ... (re-enter host, port, user, AUTO\_POSITION = 1);

START SLAVE;