# MySQL vs PostgreSQL Database Operations Report

## 1. Database Creation

* **Total Databases Created:** 100
* **Database Names:** test\_db\_1 to test\_db\_100

Each database contains a users table with a schema designed to handle large, diverse datasets. Columns are set to maximum sizes to avoid type mismatches during data loading.

| Column Name | Data Type | Description |
| --- | --- | --- |
| Name | TEXT | User’s name |
| Sex | TEXT | Gender (Male/Female) |
| Age | BIGINT | User’s age |
| Height | NUMERIC | User’s height |
| Weight | NUMERIC | User’s weight |
| Team | TEXT | Team name |
| Year | BIGINT | Year of participation |
| Season | TEXT | Season (Summer/Winter) |
| Host\_City | TEXT | Host city of the event |
| Host\_Country | TEXT | Host country of the event |
| Sport | TEXT | Sport type |
| Event | TEXT | Event name |
| GDP\_Per\_Capita\_Constant\_LCU\_Value | NUMERIC | GDP per capita (constant LCU) |
| Cereal\_yield\_kg\_per\_hectare\_Value | NUMERIC | Cereal yield (kg per hectare) |
| Military\_expenditure\_current\_LCU\_Value | NUMERIC | Military expenditure (current LCU) |
| Tax\_revenue\_current\_LCU\_Value | NUMERIC | Tax revenue (current LCU) |
| Expense\_current\_LCU\_Value | NUMERIC | Expense (current LCU) |
| Central\_government\_debt\_total\_current\_LCU\_Value | NUMERIC | Central government debt (current LCU) |
| Representing\_Host | TEXT | Whether the team represents the host |
| Avg\_Temp | TEXT | Average temperature of the host city |
| Medal | TEXT | Type of medal won (Gold/Silver/Bronze) |
| Medal\_Binary | BIGINT | Binary flag indicating if a medal was won (1/0) |

## 2. Data Insertion

* **Total Records Inserted in Each Database:** 202,616
* **Total Records Across All Databases:** 20,261,600

Each record represents a dataset row with relevant user, sports, and country data.

## 3. Disk Usage - Before Deletion

### MySQL

| Metric | Per Database Size | Total Size for 100 Databases |
| --- | --- | --- |
| Folder Size on Disk | ~50.3 MB | ~5.03 GB |
| mysql.ibd File Size | ~29.4 MB | Remained constant |

### PostgreSQL

| Metric | Per Database Size | Total Size for 100 Databases |
| --- | --- | --- |
| Folder Size on Disk | ~57.3 MB | ~5.73 GB |

## 4. Data Deletion - Removing 50% of Records

* **In the deletion operation, 50% of the records (101,308 records) were removed from each table across all databases.**

### MySQL

| Metric | Per Database Size | Total Size for 100 Databases | Percentage Change |
| --- | --- | --- | --- |
| Folder Size on Disk | ~50.3 MB | ~5.03 GB | No Change |

### PostgreSQL

| Metric | Per Database Size | Total Size for 100 Databases | Percentage Change |
| --- | --- | --- | --- |
| Folder Size on Disk | ~57.3 MB | ~5.73 GB | No Change |

## 5. Table Optimization (Reclaiming Disk Space)

After deletion, disk space was reclaimed using OPTIMIZE TABLE for MySQL and VACUUM FULL for PostgreSQL.

### MySQL Optimization

**Command Used:**

OPTIMIZE TABLE users;

| Metric | Per Database Size | Total Size for 100 Databases | Percentage Change |
| --- | --- | --- | --- |
| Folder Size on Disk | ~28.3 MB | ~2.83 GB | ↓ 43.74% |

### PostgreSQL Optimization

**Command Used:**

VACUUM FULL users;

| Metric | Per Database Size | Total Size for 100 Databases | Percentage Change |
| --- | --- | --- | --- |
| Folder Size on Disk | ~32.2 MB | ~3.22 GB | ↓ 43.80% |

## 6. Summary of Disk Usage Changes

| Operation | Per Database Size (MySQL) | Per Database Size (PostgreSQL) | MySQL % Change | PostgreSQL % Change |
| --- | --- | --- | --- | --- |
| Initial Size | ~50.3 MB | ~57.3 MB | - | - |
| After First Deletion | ~50.3 MB | ~57.3 MB | No Change | No Change |
| After Optimization | ~28.3 MB | ~32.2 MB | ↓ 43.7% | ↓ 43.80% |

## 7. Comparison Summary

| Aspect | MySQL | PostgreSQL | Winner |
| --- | --- | --- | --- |
| Disk Usage Before Delete | 5.03 GB | 5.73 GB | MySQL |
| Disk Usage After Delete | No Change | No Change | Draw |
| Disk Usage After Optimize | 2.83 GB | 3.22 GB | MySQL |
| Reclaim Efficiency (%) | 43.74% | 43.80% | PostgreSQL (by a slight margin) |

# Query Execution Plan Analysis for MySQL and PostgreSQL

### EXPLAIN in MySQL

**Purpose**:  
The EXPLAIN statement in MySQL is used to display the execution plan of a SQL query. It provides details about how the MySQL query optimizer plans to execute the query, including the order of table reads, indexes used, join methods, and more.

**How It Works**:

1. When you prefix a SELECT query with EXPLAIN, MySQL does not execute the query. Instead, it provides a breakdown of how the query will be processed.
2. The output includes:
   * **id**: The identifier for each part of the query.
   * **select\_type**: The type of query (e.g., SIMPLE, PRIMARY, SUBQUERY).
   * **table**: The table being accessed.
   * **type**: The join type or access method (e.g., ALL, index, range).
   * **possible\_keys**: The indexes that might be used.
   * **key**: The index actually used.
   * **rows**: The estimated number of rows to be examined.
   * **Extra**: Additional information, like “Using where” or “Using temporary”.

**Example**:

EXPLAIN SELECT \* FROM students WHERE age > 20;

This will display how the query accesses the students table, whether it uses an index, and how many rows it estimates to scan.

### EXPLAIN in PostgreSQL

**Purpose**:  
Like in MySQL, EXPLAIN in PostgreSQL provides a query execution plan. It shows the steps the database will take to execute the query, including join strategies, index usage, and the estimated cost of each step.

**How It Works**:

1. When you prefix a query with EXPLAIN, PostgreSQL provides an estimated execution plan without running the query.
2. The output includes:
   * **Plan type**: Indicates the operation type (e.g., Seq Scan, Index Scan, Hash Join).
   * **Cost**: Two values showing the estimated cost to start and complete the step.
   * **Rows**: The estimated number of rows returned at that step.
   * **Width**: The estimated width (in bytes) of each row.

**Example**:

EXPLAIN SELECT \* FROM students WHERE age > 20;

This will show if a sequential scan or an index scan is used, the query cost, and the number of rows expected to match.

### EXPLAIN ANALYZE in PostgreSQL

**Purpose**:  
EXPLAIN ANALYZE not only provides the execution plan but also executes the query to collect **actual runtime statistics**. This is particularly useful for comparing estimated costs to real-world performance.

**How It Works**:

1. Prefix the query with EXPLAIN ANALYZE. PostgreSQL will run the query and include actual execution details.
2. The output includes:
   * All details from EXPLAIN.
   * **Actual Time**: The actual time taken for each step (in milliseconds).
   * **Rows**: The actual number of rows processed.
   * **Loops**: The number of times a node was executed (useful in nested loops).
   * **Execution Time**: Total time taken for the query.

**Example**:

EXPLAIN ANALYZE SELECT \* FROM students WHERE age > 20;

This will provide detailed timing information for each step of the query execution.

### Key Differences

| Feature | MySQL (EXPLAIN) | PostgreSQL (EXPLAIN) | PostgreSQL (EXPLAIN ANALYZE) |
| --- | --- | --- | --- |
| Execution | No | No | Yes (query runs) |
| Estimated Costs | No | Yes | Yes |
| Actual Runtime Stats | No | No | Yes |
| Output Details | Focus on table access | Comprehensive plan steps | Comprehensive + runtime stats |

# Comparative Report: MySQL vs PostgreSQL Configuration

#### **1. Overview of Configuration Files**

| **Aspect** | **MySQL** | **PostgreSQL** |
| --- | --- | --- |
| **Main Configuration File** | my.ini (Windows) or my.cnf (Linux) | postgresql.conf |
| **Authentication File** | N/A (but can use mysql.user table for permissions) | pg\_hba.conf |
| **Other Config Files** | N/A | pg\_ident.conf |

#### **2. Server Configuration**

* **MySQL** (my.ini):
  + Defines server-specific settings, such as the default storage engine, max connections, buffer sizes, and log file locations.
  + Example:
  + [mysqld]  
    default-storage-engine=InnoDB  
    max\_connections=200  
    innodb\_buffer\_pool\_size=1G  
    log-error=/var/log/mysql/error.log
* **PostgreSQL** (postgresql.conf):
  + Similar configuration with server settings like port, memory usage, and logging.
  + Example:
  + listen\_addresses = 'localhost'  
    port = 5432  
    shared\_buffers = 128MB  
    log\_directory = '/var/log/postgresql'  
    log\_statement = 'all'

#### **3. Performance Tuning**

* **MySQL**:
  + innodb\_buffer\_pool\_size and query\_cache\_size control memory usage.
  + Example:
  + innodb\_buffer\_pool\_size=1G  
    query\_cache\_size=64M
* **PostgreSQL**:
  + shared\_buffers and work\_mem are used to allocate memory for caching and query operations.
  + Example:
  + shared\_buffers = 1GB  
    work\_mem = 4MB  
    maintenance\_work\_mem = 64MB

#### **4. Authentication and Access Control**

* **MySQL**:
  + MySQL uses access control via the mysql.user table and does not rely on an external file for authentication.
  + Connection methods (password, SSL) are controlled via SQL GRANT statements.
* **PostgreSQL**:
  + Authentication is handled by pg\_hba.conf, which defines which users can access which databases and from where.
  + Example:
  + local all postgres md5  
    host all all 192.168.1.0/24 md5

#### **5. Replication Settings**

* **MySQL**:
  + Replication settings are configured in my.ini under the [mysqld] section, including the log-bin and server-id for master-slave replication.
  + Example:
  + server-id = 1  
    log-bin = mysql-bin  
    replicate-do-db = exampledb
* **PostgreSQL**:
  + Replication is managed in postgresql.conf, with parameters like wal\_level, max\_wal\_senders, and hot\_standby for master-slave replication.
  + Example:
  + wal\_level = replica  
    max\_wal\_senders = 3  
    hot\_standby = on

#### **6. Logging Configuration**

* **MySQL**:
  + Logs are configured with options such as log-error, general\_log, and slow\_query\_log.
  + Example:
  + log-error=/var/log/mysql/error.log  
    general\_log=1  
    slow\_query\_log=1
* **PostgreSQL**:
  + Logs are configured using the log\_statement and log\_directory parameters in postgresql.conf.
  + Example:
  + log\_statement = 'all'  
    log\_directory = '/var/log/postgresql'

#### **7. Memory and Buffer Settings**

* **MySQL**:
  + Settings like innodb\_buffer\_pool\_size and key\_buffer\_size control the memory allocation for InnoDB and MyISAM engines.
  + Example:
  + innodb\_buffer\_pool\_size=1G  
    key\_buffer\_size=128M
* **PostgreSQL**:
  + Memory parameters like shared\_buffers, effective\_cache\_size, and work\_mem manage buffer and memory allocation.
  + Example:
  + shared\_buffers = 1GB  
    effective\_cache\_size = 4GB  
    work\_mem = 4MB

#### **8. Data Directories and Paths**

* **MySQL**:
  + The location of the data directory is configured in my.ini under the datadir directive.
  + Example:
  + datadir=/var/lib/mysql
* **PostgreSQL**:
  + Data directory and other paths are configured under data\_directory and temp\_tablespaces in postgresql.conf.
  + Example:
  + data\_directory = '/var/lib/postgresql/data'  
    temp\_tablespaces = '/var/lib/postgresql/tmp'

#### **9. File and Directory Locations**

* **MySQL**:
  + Log and temporary directories are defined in my.ini, and backups or binlog files can be specified.
  + Example:
  + tmpdir=/tmp
* **PostgreSQL**:
  + Similar to MySQL, PostgreSQL allows configuration of log and temporary file directories in postgresql.conf.
  + Example:
  + log\_directory = '/var/log/postgresql'  
    tmp\_tablespaces = '/tmp'