# Duplicate Removal Tool Documentation

The Duplicate Removal Tool safely identifies and removes duplicate records from your database tables. It provides multiple safety mechanisms and clear visibility into what will be removed.

## 🚀 Quick Start

# Show what duplicates exist (safe)  
npm run db:remove-duplicates --show  
  
# See what would be removed (safe)  
npm run db:remove-duplicates --dry-run  
  
# Actually remove duplicates  
npm run db:remove-duplicates --verbose

## 🎯 Command Syntax

npm run db:remove-duplicates [options]

### Options

* --show, -s - Only show duplicates, don’t remove anything
* --dry-run, -d - Show what would be removed without actually removing
* --verbose, -v - Show detailed output including removed records
* --help, -h - Show help information

## 🔍 Current Configuration

**Target Table:** users **Fields Checked:** - email - Finds users with duplicate email addresses - username - Finds users with duplicate usernames

**Removal Strategy:** - ✅ **Keeps the OLDEST record** (earliest created\_at) - 🗑️ **Removes newer duplicates** - 🔗 **Cleans up relationships** (removes user\_roles entries)

## 📖 Detailed Examples

### Example 1: Check for Duplicates (Safe)

npm run db:remove-duplicates --show

**Output:**

🧹 Duplicate User Removal Tool  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  
  
🎯 Target: Table 'users', Fields: [email, username]  
  
🔍 Finding duplicates in table: users  
📋 Checking fields: email, username  
  
 🔎 Checking email duplicates...  
 📊 Found 2 email duplicate groups  
 🔎 Checking username duplicates...  
 📊 Found 1 username duplicate groups  
  
📊 Duplicate Analysis for table: USERS  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  
  
📧 EMAIL Duplicates:  
┌─────────┬──────────────────┬───────┬─────────────┐  
│ (index) │ email │ count │ record\_ids │  
├─────────┼──────────────────┼───────┼─────────────┤  
│ 0 │ 'admin@test.com' │ 3 │ '1, 4, 7' │  
│ 1 │ 'user@test.com' │ 2 │ '2, 5' │  
└─────────┴──────────────────┴───────┴─────────────┘  
  
👤 USERNAME Duplicates:  
┌─────────┬──────────┬───────┬─────────────┐  
│ (index) │ username │ count │ record\_ids │  
├─────────┼──────────┼───────┼─────────────┤  
│ 0 │ 'admin' │ 2 │ '1, 6' │  
└─────────┴──────────┴───────┴─────────────┘

### Example 2: Dry Run (Safe Preview)

npm run db:remove-duplicates --dry-run --verbose

**Output:**

🧹 Duplicate User Removal Tool  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  
  
🎯 Target: Table 'users', Fields: [email, username]  
  
[... duplicate analysis ...]  
  
⚠️ Found duplicates! Proceeding with removal...  
🔍 DRY RUN MODE - No actual changes will be made  
  
🧹 Removing duplicates from table: users...  
  
📧 Processing email duplicates:  
  
 email: admin@test.com  
 🔒 Keeping record ID: 1 (oldest)  
 🗑️ Removing record IDs: 4, 7  
  
 email: user@test.com  
 🔒 Keeping record ID: 2 (oldest)  
 🗑️ Removing record IDs: 5  
  
👤 Processing username duplicates:  
  
 username: admin  
 🔒 Keeping record ID: 1 (oldest)  
 🗑️ Removing record IDs: 6  
  
📊 Total records to remove from users: 4  
  
🔍 DRY RUN - No changes made  
Would remove 4 duplicate records from users

### Example 3: Actual Removal

npm run db:remove-duplicates --verbose

**Output:**

🧹 Duplicate User Removal Tool  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  
  
[... duplicate analysis and removal plan ...]  
  
⚠️ Found duplicates! Proceeding with removal...  
  
🧹 Removing duplicates from table: users...  
  
[... processing details ...]  
  
📊 Total records to remove from users: 4  
  
🔗 Removing user role assignments...  
 ✅ Removed 6 role assignments  
  
👤 Removing duplicate users...  
 ✅ Removed 4 users  
  
🗑️ Removed users:  
┌─────────┬────┬─────────────────┬──────────┐  
│ (index) │ id │ email │ username │  
├─────────┼────┼─────────────────┼──────────┤  
│ 0 │ 4 │'admin@test.com' │ 'admin2' │  
│ 1 │ 5 │'user@test.com' │ 'user2' │  
│ 2 │ 6 │'test@test.com' │ 'admin' │  
│ 3 │ 7 │'admin@test.com' │ 'admin3' │  
└─────────┴────┴─────────────────┴──────────┘  
  
✅ Duplicate removal completed successfully!  
📊 Total records removed: 4  
  
💡 Run "npm run db:query users" to verify results

## 🛡️ Safety Features

### 1. Multiple Safety Levels

--show # Just look, don't touch anything  
--dry-run # Show exactly what would happen  
# (no flag) # Actually perform the removal

### 2. Smart Duplicate Detection

* **Avoids double-counting** - Won’t remove a user twice if they’re duplicate by both email AND username
* **Preserves oldest records** - Keeps the record with earliest created\_at timestamp
* **Clear identification** - Shows exactly which records will be kept vs removed

### 3. Relationship Cleanup

* **Cascading cleanup** - Removes related user\_roles entries first
* **Foreign key safe** - Handles database constraints properly
* **Transaction safety** - Operations are performed in safe order

### 4. Detailed Reporting

* **Before and after counts** - Shows exactly what changed
* **Verbose output** - Lists every record that was removed
* **Verification guidance** - Tells you how to check results

## 🔧 Troubleshooting

### No Duplicates Found

npm run db:remove-duplicates --show

**Output:**

📧 EMAIL Duplicates:  
 ✅ No email duplicates found  
  
👤 USERNAME Duplicates:  
 ✅ No username duplicates found

**Solution:** Your database is clean! No action needed.

### Permission Errors

❌ Error removing duplicates: permission denied for table users

**Solution:** Ensure your database user has DELETE permissions:

GRANT DELETE ON users, user\_roles TO your\_database\_user;

### Foreign Key Constraint Errors

❌ Error removing duplicates: foreign key constraint violation

**Solution:** The tool should handle this automatically by removing user\_roles first. If this persists, check for other tables that reference users.

### Connection Errors

❌ Database connection failed

**Solution:** 1. Check your .env file database configuration 2. Ensure database is running 3. Verify SSL settings

## 🎯 Best Practices

### 1. Always Start with –show

# Never run removal without checking first  
npm run db:remove-duplicates --show

### 2. Use Dry Run Before Actual Removal

# See exactly what will happen  
npm run db:remove-duplicates --dry-run --verbose  
  
# Then actually do it  
npm run db:remove-duplicates --verbose

### 3. Backup Before Removal

# Create database backup first  
pg\_dump your\_database > backup\_before\_duplicate\_removal.sql  
  
# Then remove duplicates  
npm run db:remove-duplicates --verbose

### 4. Verify Results

# Check that duplicates are gone  
npm run db:remove-duplicates --show  
  
# Verify remaining users look correct  
npm run db:query users

## 🔄 Common Workflows

### Regular Maintenance

# Weekly duplicate check  
npm run db:remove-duplicates --show  
  
# If duplicates found, remove them  
npm run db:remove-duplicates --dry-run  
npm run db:remove-duplicates --verbose

### After Data Import

# After importing user data  
npm run db:remove-duplicates --show  
npm run db:remove-duplicates --dry-run --verbose  
npm run db:remove-duplicates --verbose  
npm run db:query users:count

### Development Cleanup

# Clean up test data duplicates  
npm run db:remove-duplicates --show  
npm run db:remove-duplicates --verbose

## 🛠️ Customization

### Changing Target Table

Edit /api/scripts/remove-duplicates.js around line 196:

// Configuration - you can modify these  
const tableName = 'users'; // Change to other table  
const fieldsToCheck = ['email', 'username']; // Change fields

### Adding Support for Other Tables

The current version only supports the users table. To add support for other tables, you would need to:

1. Modify the removal logic in the removeDuplicates function
2. Add appropriate relationship cleanup
3. Update the schema imports

### Custom Duplicate Logic

You can modify the duplicate detection query in the findDuplicates function to use different criteria or add additional filters.

## 📊 Understanding the Output

### Duplicate Groups

* **Count**: How many records share the same value
* **Record IDs**: List of all duplicate record IDs (ordered by creation date)
* **Keep**: First ID in the list (oldest record)
* **Remove**: All other IDs in the list (newer records)

### Removal Process

1. **Analysis**: Find all duplicate groups
2. **Planning**: Determine which records to keep/remove
3. **Cleanup**: Remove related data (user\_roles)
4. **Removal**: Delete duplicate records
5. **Reporting**: Show what was removed

This tool provides a safe, transparent way to clean up duplicate data while preserving the integrity of your database relationships.