**Excluding few things**

**To exclude single directory of the same path that to change permissions of remaining directories in the same path**

cd /var/www/vhosts/staging/ && find . -type d -not -path "./wwwroot" -exec chmod 775 {} +

cd /var/www/vhosts/staging/ && find . -type d -not -path "./wwwroot" -exec chown www-data:www-data {} +

cd /var/www/vhosts/staging/

find . \( -path ./wwwroot \) -prune -o -print0 | xargs -0 chmod 775

cd /var/www/vhosts/staging/

find . \( -path ./wwwroot \) -prune -o -print0 | xargs -0 chown www-data

**To recursively give directories read & execute privileges:**

find /path/to/base/dir -type d -exec chmod 755 {} +

**To recursively give files read privileges:**

find /path/to/base/dir -type f -exec chmod 644 {} +

**Use find's -type option to limit actions to files and directories. Use the -o option to specify alternate actions for different types, so you only have to run find once, rather than separately for each type.**

find htdocs -type f -exec chmod 664 {} + -o -type d -exec chmod 775 {} +

**This example will show how to use find command to copy all files but excluding a directories from its search**

find . -type f -not -path "./dir1/\*" -not -path "./dir4/\*" -exec cp '{}' /tmp \;

Regular expressions to exclude files in the “find” command

You can use regular expressions to exclude files in the “find” command by using the “-regex” option followed by the regular expression pattern. For example, you can exclude files with names that start with “test” using the following command:

find . -type f -not -regex '.\*/test.\*'

**Back\_up & remove databases on weekly bases**

0 1 \* \* 7 bash /root/scripts/database\_backup.sh

0 1 \* \* 1 bash /root/scripts/database\_remove.sh

#! bin/bash

cd back\_up/ && mysqldump -u root -pavani@0987 --all-databases > full-backup-$(date +%F).sql

#! bin/bash

find back\_up/ -name '\*.sql' -mmin +10920 -exec rm {} \;

**Single command to rename database :-**

**Please create a new database before executing this below command**

for table in `mysql -u root -ppassword -s -N -e "use old\_db;show tables from old\_db;"`; do mysql -u root -ppassword -s -N -e "use old\_db;rename table old\_db.$table to new\_db.$table;"; done;

**Bash\_script for doing rename databases**

#!/bin/bash

# MySQL connection details

MYSQL\_USER="your\_mysql\_user"

MYSQL\_PASSWORD="your\_mysql\_password"

OLD\_DB\_NAME="old\_database\_name"

NEW\_DB\_NAME="new\_database\_name"

# Step 1: Create a new database

mysql -u "$MYSQL\_USER" -p"$MYSQL\_PASSWORD" -e "CREATE DATABASE $NEW\_DB\_NAME;"

# Step 2: Copy data from the old database to the new one

mysqldump -u "$MYSQL\_USER" -p"$MYSQL\_PASSWORD" "$OLD\_DB\_NAME" | mysql -u "$MYSQL\_USER" -p"$MYSQL\_PASSWORD" "$NEW\_DB\_NAME"

# Step 3: Optionally, drop the old database

mysql -u "$MYSQL\_USER" -p"$MYSQL\_PASSWORD" -e "DROP DATABASE $OLD\_DB\_NAME;"

echo "Database renamed successfully."