# MYSQL COMMANDS

## MYSQL FOR ANALYSTS

A quick reference for the commonly used SQL commands for Data Analysis

## mysql command-line client

Connect to MySQL server command-line client with a username and password (MySQL will prompt for a password)

```
mysql -u [username] -p;
```

## Working with databases

Create a database with a specified name if it does not exist in the database server

```
CREATE DATABASE [IF NOT EXISTS]
database_name;
```

Use a database or change the current database to another database that you are working with:

```
USE database_name;
```

Drop a database with a specified name permanently. All physical files associated with the database will be deleted.

```
DROP DATABASE [IF EXISTS]
database_name;
```

Show all available databases in the current MySQL database server

```
SHOW DATABASES;
```

## Working with tables

Show all tables in a current database.

```
SHOW TABLES;
```

Show the details of columns in a table:

```
DESCRIBE table_name;
```

# DDL Commands

(DATA DEFINITION LANGUAGE)
Work with the structure of the table.


## Create a table

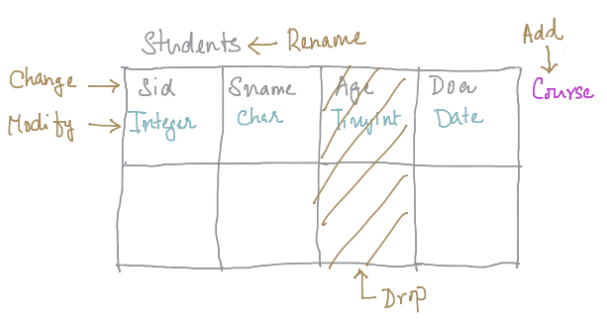
```
CREATE TABLE [IF NOT EXISTS]
table_name (
column_name data_type,
column_name data_type,
    ...
column_name data_type
);
```

Example

```
CREATE TABLE [IF NOT EXISTS]
students (
      sid integer,
      sname varchar(20),
      age tinyint,
      dob date
 );
```

## Alter table

Make modifications to the structure of the table.




Add a new column into a table:

```
ALTER TABLE table_name
ADD [COLUMN]
column_name data_type;
```

Drop a column from a table:

```
ALTER TABLE table_name
DROP [COLUMN] column_name;
```

Change the name of a column:

```
ALTER TABLE table_name
CHANGE [COLUMN]
old_column_name new_column_name
data_type;
```

Change datatype of a column:

```
ALTER TABLE table_name
MODIFY [COLUMN]
column_name new_data_type;
```

Change the name of a table:

```
ALTER TABLE table_name
RENAME TO
new_table_name;
```

## Rename table

Change the name of a table:

```
RENAME TABLE old_table_name
TO
new_table_name;
```

## Truncate table

Clear the table:

```
TRUNCATE table_name;
```

## Drop table

Drop the table:

```
DROP TABLE table_name;
```

# DML Commands

(DATA MANIPULATION LANGUAGE)
Work with the contents of the table.

## Insert Command

Add a record:
a. Insert values for all the columns in the table

```
INSERT INTO table_name
VALUES (value_list) ;
```

b. Insert values for specific columns in the table

```
INSERT INTO
table_name(column_list)
VALUES (value_list) ;
```

c. Insert multiple records

```
INSERT INTO table_name
VALUES (value_list1),
(value_list2),(value_list3);
```

## Delete Command

Delete a record:

a. Delete a record from the table

```
DELETE FROM table_name
WHERE condition;
```

a. Delete ALL records from the table

```
DELETE FROM table_name;
```

## Update Command

Make changes to a record:

a. Update a record in the table

```
UPDATE TABLE table_name
SET column = value1
[, column2 = value2,...]
WHERE condition;
```

a. Update ALL records in the table

```
UPDATE TABLE table_name
SET column = value1
[, column2 = value2,...];
```

# DQL Command

(DATA QUERY LANGUAGE)
Read data from the table.

## Select Command

a. Query all data from a table:

```
SELECT * FROM table_name;
```

b. Query data from one or more column of a table:

```
SELECT
    column1, column2, ...
FROM
    table_name;
```

c. Select unique rows from a column:

```
SELECT
    DISTINCT (column_name)
FROM
   table_name;
```

d. Query data with a filter using a WHERE clause:

```
SELECT column_list
FROM table_name
WHERE condition;
```

d. Change the display of the column name using column alias:

```
SELECT
    column1 AS alias_name,
    expression AS alias,
    ...
FROM
    table_name;
```

e. Count the number of rows in a table:

```
SELECT COUNT(*)
FROM table_name;
```

f. Sorting a result set:

```
SELECT column_list
FROM table_name
ORDER BY column1 ASC [DESC],
column2 ASC [DESC];
```

g. Group rows in a result set:

```
SELECT column_list
FROM table_name
GROUP BY column1, column2;
```

g. Filter groups rows in a result set using HAVING clause:

```
SELECT column_list
FROM table_name
GROUP BY column1
HAVING condition;
```

h. Searching data using the LIKE operator:

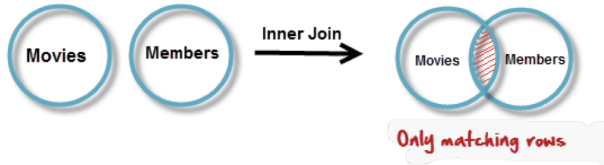
```
SELECT select_list
FROM table_name
WHERE column LIKE '%pattern%';
```

---

## Joining Tables

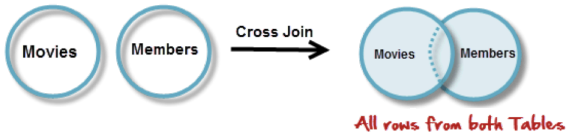
a. Query data from multiple tables using inner join:

```
SELECT table1_column_list,
table2_column_list,
FROM table1
INNER JOIN table2
ON condition;
```




d. Make a Cartesian product of rows:

```
SELECT select_list
FROM table1
CROSS JOIN table2;
```



c. Query data from multiple tables using left join:

```
SELECT table1_column_list,
table2_column_list,
FROM table1
LEFT [OUTER] JOIN table2
ON condition;
```


d. Query data from multiple tables using right join:

```
SELECT table1_column_list,
table2_column_list,
FROM table1
RIGHT [OUTER] JOIN table2
ON condition;
```


# Database Constraints

## Not Null Constraint

a. Create a not null column:

```
CREATE TABLE [IF NOT EXISTS]
table_name (
nn_column data_type NOT NULL,
column_name data_type,
    ...
column_name data_type
);
```

b. Add a not null constraint to a column :

```
ALTER TABLE table_name
MODIFY [COLUMN]
column_name data_type NOT NULL;
```

c. Remove a not null constraint from a column :

```
ALTER TABLE table_name
MODIFY [COLUMN]
nn_column data_type;
```

## Unique Constraint

a. Create a column with unique data:

```
CREATE TABLE [IF NOT EXISTS]
table_name (
unq_column data_type UNIQUE,
column_name data_type,
    ...
column_name data_type
);
```

b. Add a unique constraint to a column :

```
ALTER TABLE table_name
MODIFY [COLUMN]
column_name data_type UNIQUE;
```

c. Remove a unique constraint from a column :

```
ALTER TABLE table_name
DROP INDEX unq_column ;
```

## Check Constraint

a. Create a check constraint on a column:

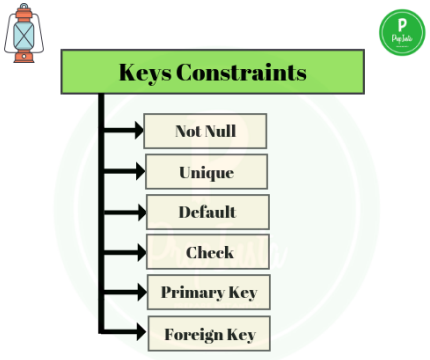
```
CREATE TABLE [IF NOT EXISTS]
table_name (
chk_column data_type
CHECK (condition),
column_name data_type,
    ...
column_name data_type
);
```

b. Add a check constraint to a column :

```
ALTER TABLE table_name
ADD CONSTRAINT [constraint_name]
CHECK (condition);
```

c. Drop a check constraint to a column :

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```




## Default Constraint

a. Create a default constraint on a column:

```
CREATE TABLE [IF NOT EXISTS]
table_name (
def_column data_type
DEFAULT default_value,
column_name data_type,
    ...
column_name data_type
);
```

b. Add a default constraint to a column :

```
ALTER TABLE table_name
MODIFY column_name data_type
DEFAULT defaut_value;
```

c. Drop a default constraint to a column :

```
ALTER TABLE table_name
ALTER def_column_name DROP
DEFAULT;
```

## Primary Key Constraint

a. Create a PRIMARY KEY constraint on a column:

```
CREATE TABLE [IF NOT EXISTS]
table_name (
pk_column data_type PRIMARY KEY,
column_name data_type,
    ...
column_name data_type
);
```

```
CREATE TABLE [IF NOT EXISTS]
table_name (
pk_column data_type,
column_name data_type,
    ...
column_name data_type,
PRIMARY KEY (pk_column)
);
```

b. Add a PRIMARY KEY constraint to a column :

```
ALTER TABLE table_name
ADD CONSTRAINT
PRIMARY KEY (column_name);
```

c. Drop a default constraint to a column :

```
ALTER TABLE table_name DROP
PRIMARY KEY;
```

# Database Objects

## Working with indexes

Create an index on a table:

```
CREATE [UNIQUE|FULLTEXT|SPATIAL]
INDEX index_name
[USING HASH|BTREE]
ON table_name (column,...);
```

a. Create a Unique index with the specified name on a table:

```
CREATE UNIQUE INDEX index_name
ON table_name (column,...);
```

b. Create a hash index with the specified name on a table:

```
CREATE INDEX index_name
USING HASH
ON table_name (column,...);
```

c. Create a Unique hash index with the specified name on a table:

```
CREATE UNIQUE INDEX index_name
USING HASH
ON table_name (column,...);
```

---

Add an index to a column :

```
ALTER TABLE table_name
ADD [UNIQUE] INDEX index_name
(column,...);
```

Drop a default constraint to a column :

```
ALTER TABLE table_name
DROP INDEX index_name;
```

---

## Working with views

Create a new view:

```
CREATE VIEW [IF NOT EXISTS]
view_name
AS
select_statement;
```

Create a new view with the WITH CHECK OPTION:

```
CREATE VIEW [IF NOT EXISTS]
view_name
AS select_statement
WITH CHECK OPTION;
```

---

Drop a view:

```
DROP VIEW [IF EXISTS] view_name;
```

Drop multiple views:

```
DROP VIEW [IF EXISTS] view1,
view2, ...;
```

---

Rename a view:

```
RENAME TABLE view_name
TO new_view_name;
```

---

## Working with sequences

Create a sequence on a table:

```
CREATE TABLE [IF NOT EXISTS]
table_name (
pk_column data_type
PRIMARY KEY AUTO_INCREMENT,
column_name data_type,
    ...);
```

Reset the initial value of a sequence:

```
ALTER TABLE table_name
AUTO_INCREMENT = new_value;
```

---