Database Management

```
# login
mysql -u username -p [database]

# clear screen
\! cls

# manage databases
CREATE DATABASE database;
USE database;
DROP DATABASE database;

# manage tables
SHOW TABLES;
DROP TABLE [IF EXISTS] table;
DESCRIBE table;
SHOW CREATE TABLE table;
EXIT
```

Removing Records

```
# delete all records
DELETE FROM table;

# delete specific records
DELETE FROM table WHERE condition;
```

Conditional Operators

AND	Logical Operator
OR	Logical Operator
NOT	Logical Operator
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal (!= sometimes accepted)
BETWEEN	Between a certain range
LIKE	Search by pattern {%, _, []}
IN	Search in multiple values
()	Nested Operators

Query Examples

```
SELECT * FROM table WHERE cost > 100;
SELECT * FROM table
WHERE col1 > 0 AND col1 < 10;
SELECT * FROM table ORDER BY col [ASC, DEC];
SELECT * FROM table WHERE (a AND b) OR (c AND
d);
SELECT * FROM table WHERE col BETWEEN 50 AND
150;
SELECT * FROM table WHERE col IN ('val1',
'val2', 'val3');
SELECT * FROM table WHERE col LIKE 'P%';
```

Inserting Records Into a Table

```
# specify columns
INSERT INTO table(col1, col2, ..., coln)
VALUES (val1, val2, ..., valn);

# detect columns
INSERT INTO table
VALUES (val1, val2, ..., valn);

# multiple values
INSERT INTO table VALUES
(val1, val2, ..., valn),
(val1, val2, ..., valn);
```

Updating Records

```
# update all records
UPDATE table_name
SET col1 = val1, col2 = val2, ..., coln = valn;

# update specific records
UPDATE table_name
SET col1 = val1, col2 = val2, ..., coln = valn
WHERE condition;
```

Data Types

```
VARCHAR(size) # Variable length string
CHAR(size) # Fixed length string
ENUM(val1, ..., valn) # Limited value string
INT[(size)] # Whole number
FLOAT[(size, p)] # Floating point number
DECIMAL[(size, p)] # Precise point number
DATE # YYYY-MM-DD
TIME # HH:MM:SS
DATETIME # YYYY-MM-DD HH:MM:SS

CAST(val AS type);

# current datetime
NOW()

#current time
CAST (NOW() AS TIME);
```

Table Alias

```
SELECT col1, col2, ..., coln
FROM table
WHERE condition
[AS] alias;
```

Query Records

```
# retrieve all records
SELECT * FROM table;

# alias columns
SELECT col1 AS alias FROM table;

# retrieve specific records
SELECT * FROM table WHERE condition;

# retrieve specific columns
SELECT col1, col2, ..., coln
FROM table
WHERE condition;

# retrieve only different values
SELECT DISTINCT col1, col2, ..., coln
FROM table
WHERE condition

# order records
# order columns can be different than select
SELECT col1, col2, ..., coln
FROM table
WHERE condition
ORDER BY col1, col2, ..., coln [ASC, DESC]

SELECT col1, col2, ..., coln
FROM table
WHERE col1 [NOT] LIKE pattern;

# null operators
SELECT * FROM table WHERE col IS NOT NULL;
SELECT * FROM table WHERE col IS NULL;

# computed columns
SELECT col1 + col2 FROM table;

# case-when-else (if-then-else)
SELECT
   CASE col1
      WHEN 'A' THEN 'Alpha'
      WHEN 'B' THEN 'Bravo'
      ELSE 'Unknown'
END AS alias
FROM table;
```

Table Alias

```
SELECT col1, col2, ..., coln
FROM table
WHERE condition
[AS] alias;
```

Limits

```
#select top N rows
SELECT * FROM table LIMIT N;

# select n rows offset by m
SELECT * FROM table LIMIT N OFFSET M;

# order by first
SELECT * FROM table
ORDER BY col
LIMIT N OFFSET M;
```

Subqueries

```
# in select clause
SELECT
   col1
   (SELECT AVG(col) FROM table2) AS alias
FROM table1

# derived/temporary table
SELECT sub1.col1, ..., sub1.coln
FROM (
   SELECT * FROM table;
) AS sub1;

# in where clause
SELECT col1, ..., coln
FROM table
WHERE val > (SELECT AVG(col) FROM table);
```

Create Table

```
# basic syntax
CREATE TABLE table (
    col1 datatype,
    col2 datatype,
    ...,
    coln datatype,
);

# example tables
CREATE TABLE Departments(
   department_id INT UNIQUE PRIMARY KEY,           # inline primary key
   department_name VARCHAR(64),
   CONSTRAINT name_unique UNIQUE (department_name) # named constraint
);

CREATE TABLE Employees(
   employee_id INT NOT NULL AUTO_INCREMENT,
   full_name VARCHAR(32) NOT NULL,
   email VARCHAR(64) UNIQUE,
   hire_date DATE NOT NULL,
   department_id INT DEFAULT 0,
   salary DECIMAL(10,2),
   PRIMARY KEY (employee_id),
   FOREIGN KEY (department_id) REFERENCES departments(department_id),
   UNIQUE (full_name),
   CHECK (email LIKE "%@%") # check constraint
);

# duplicate table
CREATE TABLE table1 AS
SELECT col1, col2, ..., coln
FROM table2
WHERE condition;
```

ALTER TABLE

```
# add primary key constraint
ALTER TABLE table
ADD [CONSTRAINT [constraint_name]]
PRIMARY KEY (col1, col2, ..., coln);

# add foreign key constraint
ALTER TABLE table1
ADD [CONSTRAINT [constraint_name]]
FOREIGN KEY (table1_col)
REFERENCES table2(table2_col);

# add unique constraint
ALTER TABLE table
ADD [CONSTRAINT [constraint_name]]
UNIQUE (col1, col2, ..., coln);

# remove constraint
ALTER TABLE table
DROP CONSTRAINT constraint_name;
```

ALTER TABLE

```
# change column data type
ALTER TABLE table
MODIFY COLUMN col datatype;

# rename column
ALTER TABLE table
RENAME col1 TO col2;

# remove column
ALTER TABLE table
DROP COLUMN col;

# rename the table
ALTER TABLE table1
rename table2
```

Aggregating Functions

```
SELECT SUM(col) FROM table;
SELECT AVG(col) FROM table;
SELECT MAX(col) FROM table;
SELECT MIN(col) FROM table;
```

Join

```
SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
[INNER] JOIN t2 ON t1.col = t2.col;

SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
LEFT JOIN t2 ON t1.col = t2.col;

SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
RIGHT JOIN t2 ON t1.col = t2.col;

SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
CROSS JOIN t2 ON t1.col = t2.col;
```

Join

```
SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
[INNER] JOIN t2 ON t1.col = t2.col;

SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
LEFT JOIN t2 ON t1.col = t2.col;

SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
RIGHT JOIN t2 ON t1.col = t2.col;

SELECT
   t1.col1, t1.col2, t2.col3
FROM table1
CROSS JOIN t2 ON t1.col = t2.col;
```

Sub Queries

```
SELECT col1, ..., coln FROM (
    SELECT
      col1, ..., coln
    FROM table
) AS t1;


SELECT AVG(sum_max) FROM (
    SELECT SUM(max_grade) AS sum_max
    FROM Assignments
    GROUP BY course_id
) AS t1;
```

Outer Join

```
SELECT
   t1.col1, t1.col2, t2.col3
FROM t1
LEFT JOIN t2 ON t1.col = t2.col;
UNION
SELECT
   t1.col1, t1.col2, t2.col3
FROM t1
RIGHT JOIN t2 ON t1.col = t2.col;
```

Count Case

```
SELECT
    COUNT(CASE WHEN col = val THEN 1 END)
AS name
FROM table;

SELECT
    col1,
    COUNT(CASE WHEN col2 = val THEN 1 END)
AS name
FROM table
GROUP BY col1;
```

Union

```
SELECT col1, ..., coln FROM table1
UNION [ALL]
SELECT col1, ..., coln FROM table2
```