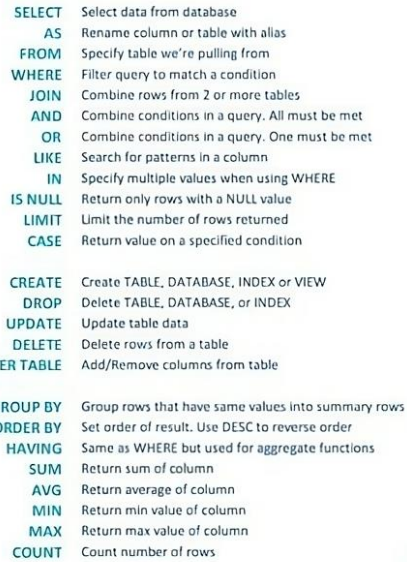
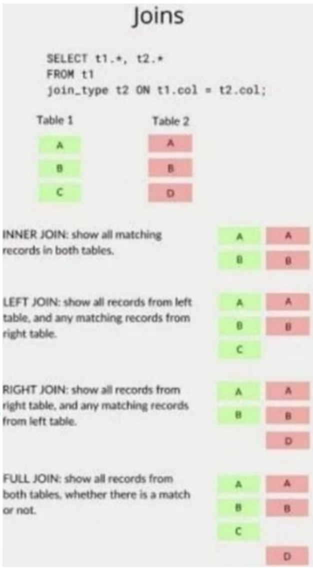
V



| | |
|---|---|
| SELECT | Select data from database |
| AS | Rename column or table with alias |
| FROM | Specify table we're pulling from |
| WHERE | Filter query to match a condition |
| JOIN | Combine rows from 2 or more tables |
| AND | Combine conditions in a query. All must be met |
| OR | Combine conditions in a query. One must be met |
| LIKE | Search for patterns in a column |
| IN | Specify multiple values when using WHERE |
| IS NULL | Return only rows with a NULL value |
| LIMIT | Limit the number of rows returned |
| CASE | Return value on a specified condition |
| CREATE | Create TABLE, DATABASE, INDEX or VIEW |
| DROP | Delete TABLE, DATABASE, or INDEX |
| UPDATE | Update table data |
| DELETE | Delete rows from a table |
| ALTER TABLE | Add/Remove columns from table |
| GROUP BY | Group rows that have same values into summary rows |
| ORDER BY | Set order of result. Use DESC to reverse order |
| HAVING | Same as WHERE but used for aggregate functions |
| SUM | Return sum of column |
| AVG | Return average of column |
| MIN | Return min value of column |
| MAX | Return max value of column |
| COUNT | Count number of rows |



## Joins

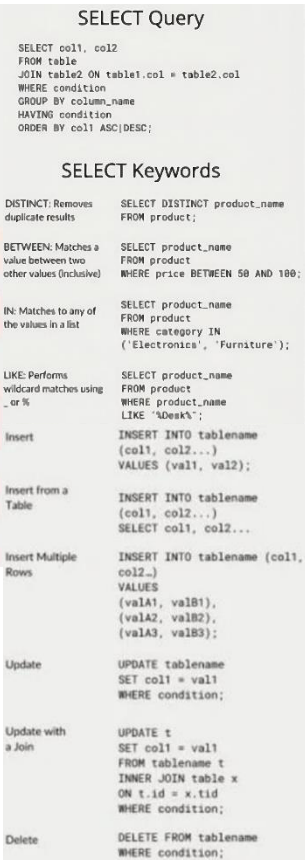
```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```

INNER JOIN: show all matching records in both tables.

LEFT JOIN: show all records from left table, and any matching records from right table.

RIGHT JOIN: show all records from right table, and any matching records from left table.

FULL JOIN: show all records from both tables, whether there is a match or not.



## SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

## SELECT Keywords

| | |
|---|---|
| DISTINCT: Removes duplicate results | `SELECT DISTINCT product_name`<br>`FROM product;` |
| BETWEEN: Matches a value between two other values (inclusive) | `SELECT product_name`<br>`FROM product`<br>`WHERE price BETWEEN 50 AND 100;` |
| IN: Matches to any of the values in a list | `SELECT product_name`<br>`FROM product`<br>`WHERE category IN`<br>`('Electronics', 'Furniture');` |
| LIKE: Performs wildcard matches using _ or % | `SELECT product_name`<br>`FROM product`<br>`WHERE product_name`<br>`LIKE '%Desk%';` |
| Insert | `INSERT INTO tablename`<br>`(col1, col2...)`<br>`VALUES (val1, val2);` |
| Insert from a Table | `INSERT INTO tablename`<br>`(col1, col2...)`<br>`SELECT col1, col2...` |
| Insert Multiple Rows | `INSERT INTO tablename (col1,`<br>`col2…)`<br>`VALUES`<br>`(valA1, valB1),`<br>`(valA2, valB2),`<br>`(valA3, valB3);` |
| Update | `UPDATE tablename`<br>`SET col1 = val1`<br>`WHERE condition;` |
| Update with a Join | `UPDATE t`<br>`SET col1 = val1`<br>`FROM tablename t`<br>`INNER JOIN table x`<br>`ON t.id = x.tid`<br>`WHERE condition;` |
| Delete | `DELETE FROM tablename`<br>`WHERE condition;` |

Restore from backup SQL File

```
mysql -u username -p dbNameYouWant < databasename_backup.sql;
```

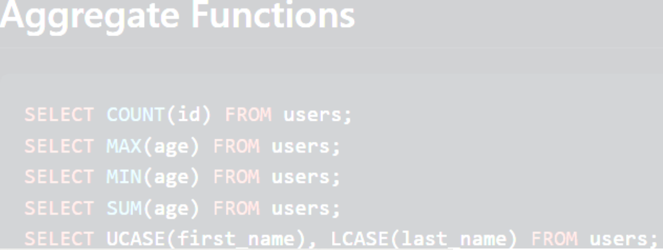
Backup Database to SQL File

```
mysqldump -u Username -p dbNameYouWant > databasename_backup.sql
```

## Users and Privileges

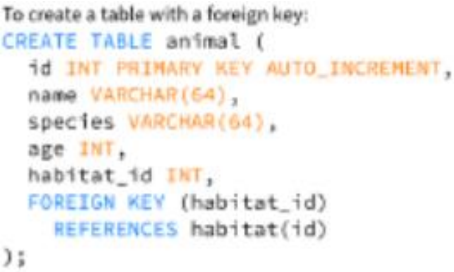
```
CREATE USER 'user'@'localhost';
GRANT ALL PRIVILEGES ON base.* TO 'user'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT, DELETE ON base.* TO 'user'@'localhost' IDENTIFIED BY 'password';
REVOKE ALL PRIVILEGES ON base.* FROM 'user'@'host'; -- one permission only
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'user'@'host'; -- all permissions
FLUSH PRIVILEGES;

SET PASSWORD = PASSWORD('new_pass');
SET PASSWORD FOR 'user'@'host' = PASSWORD('new_pass');
SET PASSWORD = OLD_PASSWORD('new_pass');

DROP USER 'user'@'host';
```

Host '%' indicates any host.



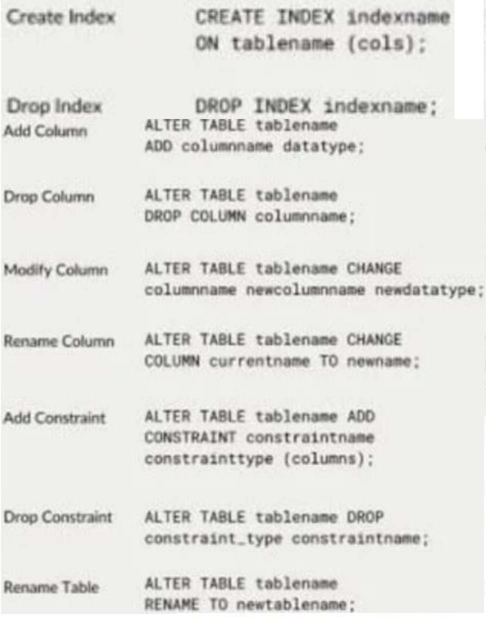
## Aggregate Functions

```
SELECT COUNT(id) FROM users;
SELECT MAX(age) FROM users;
SELECT MIN(age) FROM users;
SELECT SUM(age) FROM users;
SELECT UCASE(first_name), LCASE(last_name) FROM users;
```

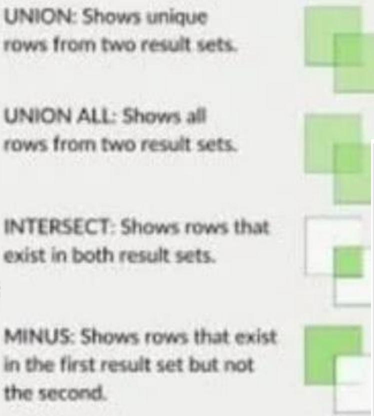


To create a table with a foreign key:

```
CREATE TABLE animal (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(64),
  species VARCHAR(64),
  age INT,
  habitat_id INT,
  FOREIGN KEY (habitat_id)
    REFERENCES habitat(id)
);
```



| | |
|---|---|
| Create Index | `CREATE INDEX indexname`<br>`ON tablename (cols);` |
| Drop Index | `DROP INDEX indexname;` |
| Add Column | `ALTER TABLE tablename`<br>`ADD columnname datatype;` |
| Drop Column | `ALTER TABLE tablename`<br>`DROP COLUMN columnname;` |
| Modify Column | `ALTER TABLE tablename CHANGE`<br>`columnname newcolumnname newdatatype;` |
| Rename Column | `ALTER TABLE tablename CHANGE`<br>`COLUMN currentname TO newname;` |
| Add Constraint | `ALTER TABLE tablename ADD`<br>`CONSTRAINT constraintname`<br>`constrainttype (columns);` |
| Drop Constraint | `ALTER TABLE tablename DROP`<br>`constraint_type constraintname;` |
| Rename Table | `ALTER TABLE tablename`<br>`RENAME TO newtablename;` |

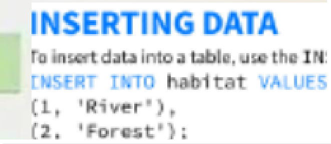


UNION: Shows unique rows from two result sets.

UNION ALL: Shows all rows from two result sets.

INTERSECT: Shows rows that exist in both result sets.

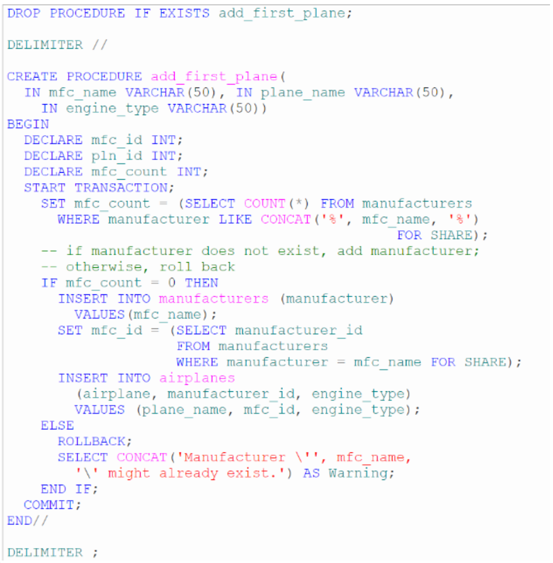
MINUS: Shows rows that exist in the first result set but not the second.



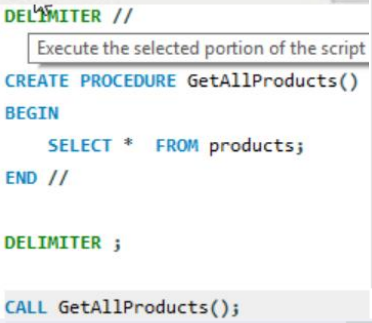
## INSERTING DATA

To insert data into a table, use the IN:

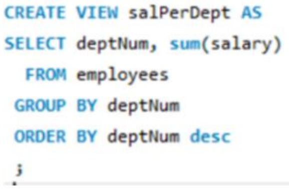
```
INSERT INTO habitat VALUES
(1, 'River'),
(2, 'Forest');
```



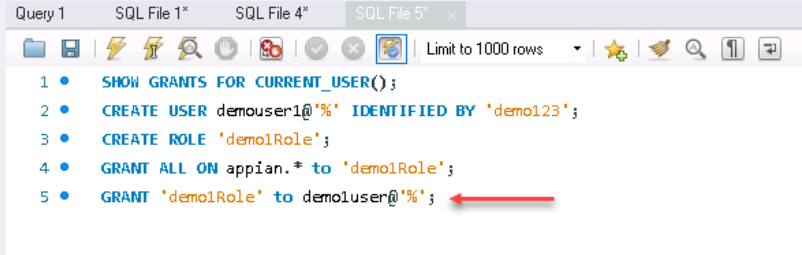
```
DROP PROCEDURE IF EXISTS add_first_plane;

DELIMITER //

CREATE PROCEDURE add_first_plane(
  IN mfc_name VARCHAR(50), IN plane_name VARCHAR(50),
    IN engine_type VARCHAR(50))
BEGIN
  DECLARE mfc_id INT;
  DECLARE pln_id INT;
  DECLARE mfc_count INT;
  START TRANSACTION;
    SET mfc_count = (SELECT COUNT(*) FROM manufacturers
      WHERE manufacturer LIKE CONCAT('%', mfc_name, '%')
                                              FOR SHARE);
    -- if manufacturer does not exist, add manufacturer;
    -- otherwise, roll back
    IF mfc_count = 0 THEN
      INSERT INTO manufacturers (manufacturer)
        VALUES(mfc_name);
      SET mfc_id = (SELECT manufacturer_id
                    FROM manufacturers
                    WHERE manufacturer = mfc_name FOR SHARE);
      INSERT INTO airplanes
        (airplane, manufacturer_id, engine_type)
        VALUES (plane_name, mfc_id, engine_type);
    ELSE
      ROLLBACK;
      SELECT CONCAT('Manufacturer \'', mfc_name,
        '\' might already exist.') AS Warning;
    END IF;
  COMMIT;
END//

DELIMITER ;
```



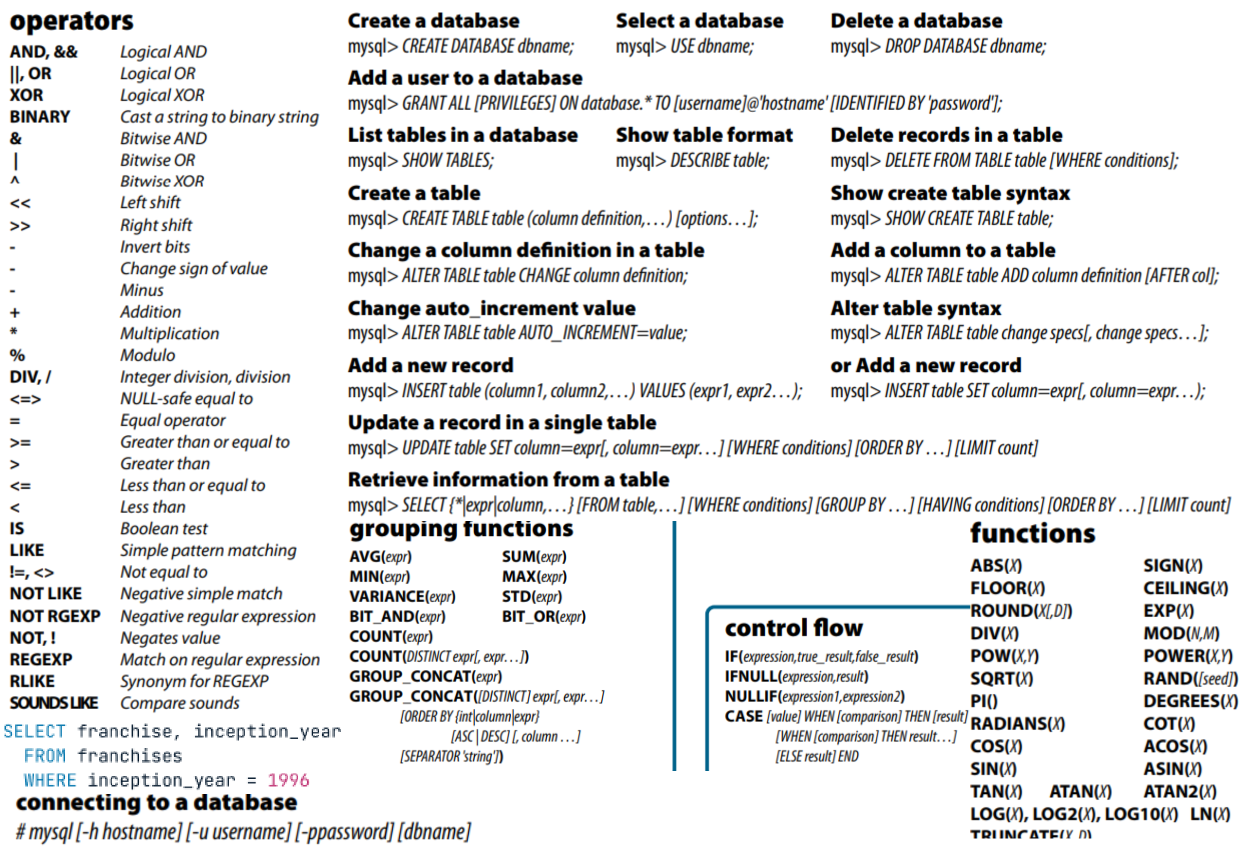
```
DELIMITER //

CREATE PROCEDURE GetAllProducts()
BEGIN
    SELECT *  FROM products;
END //

DELIMITER ;

CALL GetAllProducts();
```



```
CREATE VIEW salPerDept AS
SELECT deptNum, sum(salary)
  FROM employees
  GROUP BY deptNum
  ORDER BY deptNum desc
  ;
```



```
SHOW GRANTS FOR CURRENT_USER();
CREATE USER demouser1@'%' IDENTIFIED BY 'demo123';
CREATE ROLE 'demo1Role';
GRANT ALL ON appian.* to 'demo1Role';
GRANT 'demo1Role' to demo1user@'%';
```



## operators

| | |
|---|---|
| **AND, &&** | *Logical AND* |
| **\|\|, OR** | *Logical OR* |
| **XOR** | *Logical XOR* |
| **BINARY** | *Cast a string to binary string* |
| **&** | *Bitwise AND* |
| **\|** | *Bitwise OR* |
| **^** | *Bitwise XOR* |
| **<<** | *Left shift* |
| **>>** | *Right shift* |
| **-** | *Invert bits* |
| **-** | *Change sign of value* |
| **-** | *Minus* |
| **+** | *Addition* |
| ***** | *Multiplication* |
| **%** | *Modulo* |
| **DIV, /** | *Integer division, division* |
| **<=>** | *NULL-safe equal to* |
| **=** | *Equal operator* |
| **>=** | *Greater than or equal to* |
| **>** | *Greater than* |
| **<=** | *Less than or equal to* |
| **<** | *Less than* |
| **IS** | *Boolean test* |
| **LIKE** | *Simple pattern matching* |
| **!=, <>** | *Not equal to* |
| **NOT LIKE** | *Negative simple match* |
| **NOT RGEXP** | *Negative regular expression* |
| **NOT, !** | *Negates value* |
| **REGEXP** | *Match on regular expression* |
| **RLIKE** | *Synonym for REGEXP* |
| **SOUNDS LIKE** | *Compare sounds* |

```
SELECT franchise, inception_year
  FROM franchises
  WHERE inception_year = 1996
```

### Create a database

```
mysql> CREATE DATABASE dbname;
```

### Select a database

```
mysql> USE dbname;
```

### Delete a database

```
mysql> DROP DATABASE dbname;
```

### Add a user to a database

```
mysql> GRANT ALL [PRIVILEGES] ON database.* TO [username]@'hostname' [IDENTIFIED BY 'password'];
```

### List tables in a database

```
mysql> SHOW TABLES;
```

### Show table format

```
mysql> DESCRIBE table;
```

### Delete records in a table

```
mysql> DELETE FROM TABLE table [WHERE conditions];
```

### Create a table

```
mysql> CREATE TABLE table (column definition,...) [options...];
```

### Show create table syntax

```
mysql> SHOW CREATE TABLE table;
```

### Change a column definition in a table

```
mysql> ALTER TABLE table CHANGE column definition;
```

### Add a column to a table

```
mysql> ALTER TABLE table ADD column definition [AFTER col];
```

### Change auto_increment value

```
mysql> ALTER TABLE table AUTO_INCREMENT=value;
```

### Alter table syntax

```
mysql> ALTER TABLE table change specs[, change specs...];
```

### Add a new record

```
mysql> INSERT table (column1, column2,...) VALUES (expr1, expr2...);
```

### or Add a new record

```
mysql> INSERT table SET column=expr[, column=expr...);
```

### Update a record in a single table

```
mysql> UPDATE table SET column=expr[, column=expr...] [WHERE conditions] [ORDER BY ...] [LIMIT count]
```

### Retrieve information from a table

```
mysql> SELECT {*|expr|column,...} [FROM table,...] [WHERE conditions] [GROUP BY ...] [HAVING conditions] [ORDER BY ...] [LIMIT count]
```

## grouping functions

**AVG**(*expr*) **SUM**(*expr*)
**MIN**(*expr*) **MAX**(*expr*)
**VARIANCE**(*expr*) **STD**(*expr*)
**BIT_AND**(*expr*) **BIT_OR**(*expr*)
**COUNT**(*expr*)
**COUNT**(*DISTINCT expr[, expr...]*)
**GROUP_CONCAT**(*expr*)
**GROUP_CONCAT**(*[DISTINCT] expr[, expr...]*
*[ORDER BY {int|column|expr}*
*[ASC | DESC] [, column ...]*
*[SEPARATOR 'string']*)

## control flow

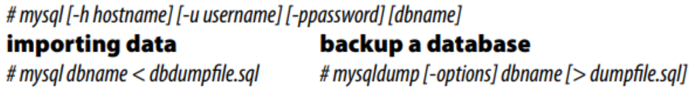
**IF**(*expression,true_result,false_result*)
**IFNULL**(*expression,result*)
**NULLIF**(*expression1,expression2*)
**CASE** *[value] WHEN [comparison] THEN [result]*
*[WHEN [comparison] THEN result...]*
*[ELSE result] END*

## functions

**ABS**(*X*) **SIGN**(*X*)
**FLOOR**(*X*) **CEILING**(*X*)
**ROUND**(*X[,D]*) **EXP**(*X*)
**DIV**(*X*) **MOD**(*N,M*)
**POW**(*X,Y*) **POWER**(*X,Y*)
**SQRT**(*X*) **RAND**(*[seed]*)
**PI**() **DEGREES**(*X*)
**RADIANS**(*X*) **COT**(*X*)
**COS**(*X*) **ACOS**(*X*)
**SIN**(*X*) **ASIN**(*X*)
**TAN**(*X*) **ATAN**(*X*) **ATAN2**(*X*)
**LOG**(*X*), **LOG2**(*X*), **LOG10**(*X*) **LN**(*X*)
**TRUNCATE**(*X,D*)

### connecting to a database

```
# mysql [-h hostname] [-u username] [-ppassword] [dbname]
```

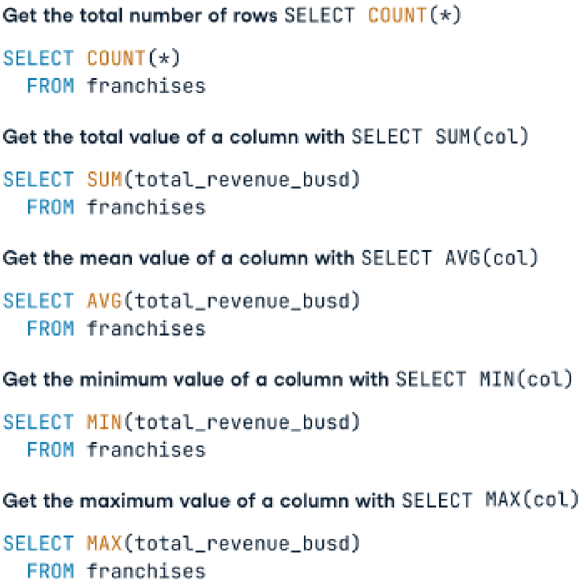


### importing data

```
# mysql dbname < dbdumpfile.sql
```

### backup a database

```
# mysqldump [-options] dbname [> dumpfile.sql]
```



Get the total number of rows `SELECT COUNT(*)`

```
SELECT COUNT(*)
  FROM franchises
```

Get the total value of a column with `SELECT SUM(col)`

```
SELECT SUM(total_revenue_busd)
  FROM franchises
```

Get the mean value of a column with `SELECT AVG(col)`

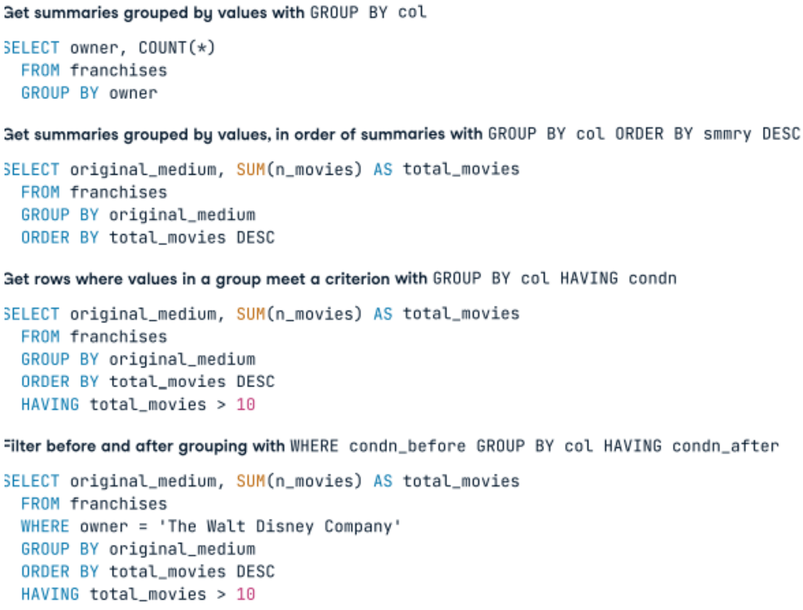
```
SELECT AVG(total_revenue_busd)
  FROM franchises
```

Get the minimum value of a column with `SELECT MIN(col)`

```
SELECT MIN(total_revenue_busd)
  FROM franchises
```

Get the maximum value of a column with `SELECT MAX(col)`

```
SELECT MAX(total_revenue_busd)
  FROM franchises
```



Get summaries grouped by values with `GROUP BY col`

```
SELECT owner, COUNT(*)
  FROM franchises
  GROUP BY owner
```

Get summaries grouped by values, in order of summaries with `GROUP BY col ORDER BY smmry DESC`

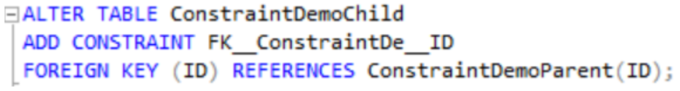
```
SELECT original_medium, SUM(n_movies) AS total_movies
  FROM franchises
  GROUP BY original_medium
  ORDER BY total_movies DESC
```

Get rows where values in a group meet a criterion with `GROUP BY col HAVING condn`

```
SELECT original_medium, SUM(n_movies) AS total_movies
  FROM franchises
  GROUP BY original_medium
  ORDER BY total_movies DESC
  HAVING total_movies > 10
```

Filter before and after grouping with `WHERE condn_before GROUP BY col HAVING condn_after`

```
SELECT original_medium, SUM(n_movies) AS total_movies
  FROM franchises
  WHERE owner = 'The Walt Disney Company'
  GROUP BY original_medium
  ORDER BY total_movies DESC
  HAVING total_movies > 10
```



```
ALTER TABLE ConstraintDemoChild
ADD CONSTRAINT FK__ConstraintDe__ID
FOREIGN KEY (ID) REFERENCES ConstraintDemoParent(ID);
```