Kurzreferenz SQL

CREATE DATABASE
```
CREATE [OR REPLACE] DATABASE [IF NOT EXISTS] db_name
```

DROP DATABASE
```
DROP DATABASE [IF EXISTS] db_name
```

USERS/PERMISSIONS
```
CREATE [OR REPLACE] USER [IF NOT EXISTS] 'user_name' IDENTIFIED BY 'password'
GRANT ALL ON database.* TO 'user_name'
REVOKE ALL ON database.* FROM 'user_name'
DROP USER [IF EXISTS] 'user_name'
```

LOAD DATA
```
LOAD DATA LOCAL INFILE 'file_name' INTO TABLE tbl_name [FIELDS TERMINATED BY 'string'] [ENCLOSED
BY 'string'] [LINES TERMINATED BY 'string'] [IGNORE number ROWS]
```

SOURCE
```
source file_name
```

CREATE TABLE
```
CREATE [OR REPLACE] TABLE [IF NOT EXISTS] tbl_name
    (col_name datatype [NOT NULL | NULL] [DEFAULT default_value | (expression)]
    [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY], ... | FOREIGN KEY (index_col_name, ... )
REFERENCES tbl_name (index_col_name, ... ) | CONSTRAINT [constraint_name] CHECK (expression))
CREATE [OR REPLACE] TABLE [IF NOT EXISTS] tbl_name LIKE old_table_name
```

ALTER TABLE
```
ALTER TABLE tbl_name [ADD | MODIFY | DROP]
    (col_name [NOT NULL | NULL] [DEFAULT default_value | (expression)]
    [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY], ... | FOREIGN KEY (index_col_name, ... )
REFERENCES tbl_name (index_col_name, ... ) | CONSTRAINT [constraint_name] CHECK (expression))
```

DROP TABLE
```
DROP TABLE [IF EXISTS] tbl_name
```

SELECT
```
SELECT [ALL | DISTINCT] select_expr [, select_expr ...]
  [FROM table_reference
    [[INNER | {LEFT|RIGHT} OUTER] JOIN table_reference ON join_condition]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position} [ASC | DESC], ...]
    [ORDER BY {col_name | expr | position} [ASC | DESC], ...]]
```

VIEWS
```
CREATE [OR REPLACE] VIEW [IF NOT EXISTS] view_name [(column_list)] AS select_statement
ALTER VIEW view_name [(column_list)] AS select_statement
DROP VIEW [IF EXISTS] view_name [, view_name] ...
```

INDEX/UNIQUE INDEX
```
CREATE [OR REPLACE] [UNIQUE] INDEX [IF NOT EXISTS] index_name ON tbl_name (index_col_name, ...)
```

INSERT
```
INSERT INTO tbl_name [(col,...)] VALUES ({expr | DEFAULT},...),(...),...
      [, select_expr ...]]
```

UPDATE
```
UPDATE table_reference
  SET col1={expr1|DEFAULT} [,col2={expr2|DEFAULT}] ...
  [WHERE where_condition]
```

DELETE
```
DELETE FROM tbl_name
    [WHERE where_condition]
```

PREPARED STATEMENTS
```
PREPEARE stmt_name FROM preparable_stmt
EXECUTE stmt_name [USING expression, ...]
```

TRANSACTIONS
```
START TRANSACTION | BEGIN [WORK]
COMMIT [WORK]
ROLLBACK [WORK]
```

STORED PROCEDURE
```
DELIMITER //
CREATE [OR REPLACE] PROCEDURE [IF NOT EXISTS] sp_name ([proc_parameter[, ...]])
BEGIN
  ...
END//
DELIMITER ;
proc_parameter: [IN | OUT | INOUT] param_name datatype
CALL sp_name([param_1[, ...]])
```

FUNCTION
```
DELIMITER //
CREATE [OR REPLACE] FUNCTION [IF NOT EXISTS] func_name ([func_parameter[, ...]]) RETURNS
datatype
BEGIN
  ...
END//
DELIMITER ;
func_parameter: param_name datatype
```

TRIGGER
```
DELIMITER //
CREATE [OR REPLACE] TRIGGER [IF NOT EXISTS] trigger_name [BEFORE | AFTER] [INSERT | UPDATE |
DELETE] ON tbl_name FOR EACH ROW
BEGIN
  ...
END//
DELIMITER ;
```

STATEMENTS
```
DECLARE var_name datatype [DEFAULT value]
SET var_name = value
IF condition THEN statements [ELSE | ELSEIF condition THEN statements] END IF
CASE expression WHEN value_1 THEN statements [WHEN value_2 THEN statements | ELSE statements]
END CASE
CASE WHEN condition_1 THEN statements [WHEN condition_2 THEN statements | ELSE statements] END
CASE
WHILE condition DO statements END WHILE
REPEAT statements UNTIL condition END REPEAT
FOR var_name IN [REVERSE] lower_bound .. upper_bound DO statements END FOR
RETURN value
```

DATATYPES
```
INT Ganzzahliger Wert
DECIMAL(x, y) Fließkommazahl mit x Stellen, davon y Nachkommastellen
CHAR(x) Zeichenkette mit x Zeichen
VARCHAR(x) Variable Zeichenkette mit maximal x Zeichen
DATE Datumswert im Format yyyy-mm-dd
DATETIME Datums-/Zeitwert im String-Format "yyyy-mm-dd hh:nn:ss"
```

AGGREGAT FUNCTIONS
```
COUNT() Anzahl
SUM() Summe
MIN() minimaler Wert
MAX() maximaler Wert
AVG() durchschnittlicher Wert
```

BUILT-IN FUNCTIONS
```
YEAR(datevalue) Jahr eines Datum-/Zeitwerts extrahieren
MONTH(datevalue) Monat eines Datum-/Zeitwerts extrahieren
ROUND(numvalue, precision) numerische Zahl auf Nachkommastelle runden
CONCAT(string1, string2, ...) Strings verketten
```