SQL CHEAT SHEET

QUERYING DATA FROM A TABLE

SELECT c1, c2 FROM t;
Query data in columns c1, c2 from a table

SELECT * FROM t;
Query all rows and columns from a table

SELECT c1, c2 FROM t
WHERE condition;
Query data and filter rows with a condition

SELECT DISTINCT c1 FROM t
WHERE condition;
Query distinct rows from a table

SELECT c1, c2 FROM t
ORDER BY c1ASC [DESC];
Sort the result setin ascending or descending order

SELECT c1, c2 FROM t
ORDER BY c1
LIMIT nOFFSET offset;
Skip *offset*of rows and return the next n rows

SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
Group rows using an aggregate function

SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
Filter groups using HAVING clause

QUERYING FROM MULTIPLE

TABLES SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
Inner join t1 and t2

SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
Left join t1 and t1

SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
Right join t1 and t2

SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
Perform full outer join

SELECT c1, c2
FROM t1
CROSS JOIN t2;
Produce a Cartesian product of rows in tables

SELECT c1, c2
FROM t1, t2;
Another way to perform cross join

SELECT c1, c2
FROM t1 A
INNER JOIN t2 BON condition;
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
Combine rows from two queries

SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
Return the intersection of two queries

SELECT c1, c2 FROM t1
MINUS
SELECTc1, c2 FROM t2;
Subtract a result set from another result set

SELECT c1, c2 FROM t1
WHERE c1[NOT] LIKE pattern;
Query rows using pattern matching %, _

SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
Query rows in a list

SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND high;
Query rows between two values

SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;
Check if values in a table is NULL or not

SQL CHEAT SHEET

MANAGING TABLES

```
CREATE TABLE t (
idINT PRIMARY KEY,
        nameVARCHAR NOT NULL,
priceINT DEFAULT 0
);
```
Createa new table with three columns

```
DROP TABLE t ;
```
Delete the table from the database

```
ALTER TABLE t ADDcolumn;
```
Add a new column to the table

```
ALTER TABLE t DROP COLUMN c ;
```
Drop column c from the table

```
ALTER TABLE t ADD constraint;
```
Add a constraint

```
ALTER TABLE t DROP constraint;
```
Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```
Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1TO c2;
```
Rename column c1 to c2

```
TRUNCATE TABLE t;
```
Remove all data in a table

USING SQL CONSTRAINTS

```
CREATE TABLE t(
     c1INT, c2INT, c3VARCHAR,
PRIMARY KEY (c1,c2)
);
```
Set c1 and c2 as a primary key

```
CREATE TABLE t1(
c1INT PRIMARY KEY,
c2INT,
   FOREIGN KEY (c2)REFERENCES t2(c2)
);
```
Set c2 column as a foreign key

```
CREATE TABLE t(
c1INT, c1INT,
UNIQUE(c2,c3)
);
```
Make the valuesin c1 and c2 unique

```
CREATE TABLE t(
c1INT, c2INT,
CHECK(c1> 0 AND c1 >= c2)
);
```
Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
c1INT PRIMARY KEY,
c2VARCHAR NOT NULL
);
```
Set values in c2 column not NULL

MODIFYING DATA

```
INSERT                    INTO
t(column_list)
VALUES(value_list);
```
Insert one row into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
(value_list), ….;
```
Insert multiple rows into a table

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

```
UPDATE t
SET c1= new_value;
```
Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
c2 = new_value
WHERE condition;
```
Update values in the column c1, c2that match the condition

```
DELETE FROM t;
```
Delete all data in a table

```
DELETE FROM t
WHERE condition;
```
Deletesubset of rows in a table

SQL CHEAT SHEET

MANAGING VIEWS

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
```
Createa new view that consists of c1 and c2

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
WITH [CASCADED | LOCAL] CHECK OPTION;
```
Create a new view with check option

```
CREATE RECURSIVEVIEW v
AS
select-statement--anchor part
UNION [ALL]
select-statement;--recursive part
```
Create a recursive view

```
CREATE  TEMPORARYVIEW  v
AS
SELECT c1, c2
FROM t;
```
Create a temporary view

DROP VIEW;
view_name Delete a
view

MANAGING INDEXES

```
CREATE INDEXidx_name
ONt(c1,c2);
```
Create an index on c1 and c2 of the table t

```
CREATE UNIQUE INDEXidx_name
ONt(c3,c4);
```
Create a unique index on c3, c4 of the table t

DROP INDEX
idx_name; Drop an
index

SQL AGGREGATE FUNCTIONS

AVGreturns the average of a list

COUNTreturns the number of elements of a list

SUMreturns the total of a list

MAXreturns the maximum value in a list

MINreturns the minimum value in a list

MANAGING TRIGGERS

```
CREATE OR MODIFY TRIGGER trigger_name
WHEN EVENT
ON table_nameTRIGGER_TYPE
EXECUTE stored_procedure;
```
Create ormodifya trigger

WHEN
- **BEFORE** –invokebefore the event occurs
- **AFTER** –invokeafter the event occurs

EVENT
- **INSERT** –invokefor INSERT
- **UPDATE** –invokefor UPDATE
- **DELETE** –invokefor DELETE

TRIGGER_TYPE
- **FOR EACH ROW**
- **FOR EACH STATEMENT**

```
CREATE TRIGGER before_insert_person
BEFORE INSERT
ON person FOR EACH ROW
EXECUTE stored_procedure;
```
Create a trigger invoked before a new row is inserted into the person table

DROP TRIGGER;
trigger_name Delete a
specific trigger