SQL_Cheat_Sheet_Mohamed_Ahmed Cheat Sheet

by MNMO via cheatography.com/135369/cs/28078/

Data Base

CREATE DATABASE DatabaseName;

DROP DATABASE DatabaseName;

SHOW DATABASES;

USE DatabaseName;

Create & Delete Table

Create Table:

CREATE TABLE table_name(

column1 datatype,

column3 datatype,

.....

columnN datatype,

PRIMARY KEY(one or more columns));

```
SQL> CREATE TABLE CUSTOMERS(
ID          INT              NOT NULL,
NAME        VARCHAR (20)     NOT NULL,
AGE         INT              NOT NULL,
ADDRESS     CHAR (25) ,
SALARY      DECIMAL (18, 2),
PRIMARY KEY (ID));
```

Creating a Table from an Existing Table:

CREATE TABLE NEW_TABLE_NAME AS

SELECT [column1, column2...columnN]

FROM EXISTING_TABLE_NAME

[WHERE]

Create & Delete Table (cont)

DROP or DELETE Table:

DROP TABLE table_name;

Constraints

```
CREATE TABLE CUSTOMERS(
ID         INT               NOT NULL,
NAME       VARCHAR (20)      NOT NULL,
AGE        INT               NOT NULL UNIQUE,
ADDRESS    CHAR (25),
SALARY     DECIMAL (18, 2) DEFAULT 5000.00,
PRIMARY KEY (ID) );
```

Appling Constraints By:

ALTER TABLE Table_Name Column MODIFY CONSTRAINT;

Dropping Constraints By:

ALTER TABLE Table_Name Column DROP CONSTRAINT;

• NOT NULL Constraint: Ensures that a column cannot have a NULL value.

You must use the IS NULL or IS NOT NULL operators to check for a NULL value.

```
SQL> SELECT
ID, NAME, AGE, ADDRESS, SALARY
FROM CUSTOMERS
WHERE SALARY IS NOT NULL;
```

Constraints (cont)

• DEFAULT Constraint: Provides a default value for a column when none is specified.

• UNIQUE Constraint: Ensures that all values in a column are different.

• PRIMARY Key: Uniquely identifies each row/record in a database table.

• FOREIGN Key: Uniquely identifies row/record in any of the given database tables. The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

• CHECK Constraint: The CHECK constraint ensures that all the values in a column satisfies certain conditions.

• INDEX: Used to create and retrieve data from the database very quickly. it is assigned a ROWID for each row before it sorts out the data.

CREATE INDEX index_name

ON table_name (column1, column2.....);

Constraints

```
CREATE TABLE CUSTOMERS(
ID INT NOT NULL,
NAME VARCHAR (20) NOT
NULL,
AGE INT NOT NULL
UNIQUE,
ADDRESS CHAR (25),
SALARY DECIMAL (18, 2)
DEFAULT 5000.00,
PRIMARY KEY (ID) );
```

Query's for Manipulating Tables

INSERT:

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)]

VALUES (value1, value2, value3,...valueN);

SELECT:

SELECT column1, column2, columnN FROM table_name;

UPDATE:

UPDATE table_name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

DELETE:

DELETE FROM table_name

WHERE [condition];

ORDER BY Clause & SORTING Results

ascending or descending order, ascending order by default.

SELECT column-list

FROM table_name

[WHERE condition]

[ORDER BY column1, column2, .. columnN] [ASC | DESC];

TOP, LIMIT or ROWNUM Clause

SELECT TOP number|percent column_name(s)

FROM table_name

WHERE [condition]

```
SQL> SELECT TOP 3 * FROM CUSTOMERS;
SQL> SELECT * FROM CUSTOMERS LIMIT 3;
SQL> SELECT * FROM CUSTOMERS WHERE ROWNUM <= 3;
```

WHERE Clause

SELECT column1, column2, column

FROM table_name

WHERE [condition]

You can specify a condition using the comparison or logical operators like >, <, =, LIKE, NOT,AND,OR.

The AND | OR Operator

SELECT column1, column2, column

FROM table_name

WHERE [condition1] AND | OR [condition2]...AND | OR [conditionN];

LIKE | Wildcard

• The percent sign (%)

• The underscore (_)

SELECT FROM table_name

WHERE column [LIKE | Wildcard] ['XXXX%' | '%XXXX%' | 'XXXX_' | '_XXXX' | '_XXXX_']

GROUP BY

SELECT column1, column2

FROM table_name

WHERE [conditions]

GROUP BY column1, column2

ORDER BY column1, column2

HAVING Clause

SELECT column1, column2

FROM table1, table2

WHERE [conditions]

GROUP BY column1, column2

HAVING [conditions]

ORDER BY column1, column2

Distinct Keyword

SELECT DISTINCT column1, column2,.....columnN

FROM table_name

WHERE [condition]

UNION | UNION ALL | INTERSECT | EXCEPT

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

To use this UNION clause, each SELECT statement must have

• The same number of columns selected

• The same number of column expressions

• The same data type

• Have them in the same order

SELECT column1 [, column2]

FROM table1 [, table2] [WHERE condition]

[UNION | UNION ALL | INTERSECT | EXCEPT]

SELECT column1 [, column2]

FROM table1 [, table2] [WHERE condition]

Joins

There are different types of joins available in SQL:

• INNER JOIN: returns rows when t is a match in both tables.

• INNER JOIN: returns rows when t is a match in both tables.

• RIGHT JOIN: returns all rows fror right table, even if there are no matches in the left table.

• FULL JOIN: returns rows when th is a match in one of the tables.

SELECT table1.column1, table2.co lumn2... FROM table1

[INNER JOIN | LEFT JOIN | RIGHT JOIN | FULL JOIN] table2

ON table1.common_field = table2.c mmon_field;

• SELF JOIN: is used to join a table itself as if the table were two tables temporarily renaming at least one t in the SQL statement.

```
SQL> SELECT
a.ID, b.NAME, a.SALARY
FROM CUSTOMERS a, CUSTOMER
WHERE a.SALARY < b.SALARY;
```

Alias

The basic syntax of a table alias

SELECT column1, column2.... FROM table_name AS alias_name WHERE [condition];

The basic syntax of a column alias

SELECT column_name AS alias_name FROM table_name WHERE [condition];

Indexes

CREATE INDEX index_name ON table_name;

Single-Column Indexes

CREATE INDEX index_name ON table_name (column_name);

Unique Indexes

CREATE UNIQUE INDEX index_name on table_name (column_name);

DROP INDEX

DROP INDEX index_name;

When should indexes be avoided?
The following guidelines indicate when the use of an index should be reconsidered.
•Indexes should not be used on small tables.
•Tables that have frequent, large batch updates or insert operations.
•Indexes should not be used on columns that contain a high number of NULL values.
•Columns that are frequently manipulated should not be indexed.

Using Views

which are a type of virtual tables allow users to do the following:

• Structure data in a way that users or classes of users find natural or intuitive.
• Restrict access to the data in such a way that a user can see and (somet imes) modify exactly what they need and no more.
• Summarize data from various tables which can be used to generate reports.

CREATE VIEW

```
CREATE VIEW view_name AS
SELECT column1, column -
2.....
FROM table_name
WHERE [condi tion];
```

Dropping Views

```
DROP VIEW view_name;
```

Transactions

Transactions have the following four standard proper ties, usually referred to by the acronym ACID.
• Atomicity: ensures that all operations within the work unit are completed succes sfully. Otherwise, the transa - ction is aborted at the point of failure and all the previous operations are rolled back to their former state.
• Consis tency: ensures that the database properly changes states upon a succes - sfully committed transa ction.
• Isolation: enables transa ctions to operate indepe ndently of and transp arent to each other.
• Durabi lity: ensures that the result or effect of a committed

Transactions (cont)

> transaction persists in case of a system failure.

Transaction Control

The following commands are used to control transactions.

• COMMIT: to save the changes.

COMMIT;

• ROLLBACK: to roll back the changes.

ROLLBACK;

• SAVEPOINT: creates points within the groups of transactions in which to ROLLBACK.

SAVEPOINT SAVEPOINT-_NAME;

ROLLBACK TO SAVEPOINT-_NAME;

• SET TRANSACTION: Places a name on a transaction.

SET TRANSACTION [READ WRITE | READ ONLY];

• The RELEASE SAVEPOINT Command

RELEASE SAVEPOINT SAVEPOINT_NAME;