CREATE TABLE table_name

DELETE FROM customers;

SQL
CHEAT SHEET

SCALER Topics

SELECT distinct(name)

WHERE condition

DATA Related Commands

CREATE TABLE

SQL command used to create a new table in a database.

Syntax:

```
CREATE TABLE table_name
(
      column_name1 data_type,
      column_name2 data_type,
      column_name3 data_type,
      ...
)
```

Examples:

Create a table called “customers” with columns for

customer_id,
customer_name,
contact_name, and
an integer field called age:

```
CREATE TABLE
  customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(50),
    contact_name VARCHAR(50),
    age INT
  );
```

INSERT INFO

SQL command used to create a new table in a database.

Syntax:

```
INSERT INTO table_name(column1,
column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Examples:

Insert a new row in the customer table created above.

```
INSERT INTO customers(customer_id,
customer_name, contact_name, age)
VALUES (1, 'John Smith', 'Jane Doe', 30);
```

ALTER TABLE

SQL command that is used to modify the structure of a table, such as adding or deleting columns.

Syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

```
ALTER TABLE table_name
RENAME COLUMN old_name to new_name;
```

OR,

```
ALTER TABLE table_name
DROP COLUMN column_name
```

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Examples:

Change the data type of the "age" column in your "customers" table from INT to BIGINT

```
ALTER TABLE customers MODIFY age BIGINT;
```

DROP TABLE Command

DROP TABLE Command is used to entire the table itself, along with its contents.

Syntax:

```
DROP TABLE table_name;
```

Examples:

DROP TABLE customers;

DELETE Command

DELETE Command is used to entire the contents of the table only, not the table

Syntax:

```
DELETE FROM table_name;
```

Examples:

DELETE FROM customers;

Basics

Select Clause

It is a SQL clause that specifies which columns to retrieve from a database table.

Syntax:

```
SELECT column1, column2, ...
FROM table_name;
```

Examples:

```
SELECT
  *
FROM
  Course;
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
202	ASE	ECE	3
211	FD	ME	3
213	DSP	SIGCOM	3
222	BCS	EE	4
301	RC	CY	4
303	MET	ME	3
323	SE	IT	4
506	DM	MS	4
511	OPT	PH	3
518	ORG	CH	4
523	NEO	BIO	4
604	WLD	CY	3
655	CNS	CSE	3
702	MIB	BIO	4
710	ECM	VLSI	3
716	SPT	SIGCOM	4
722	TBS	CSE	4

```
SELECT
  CourseID,
  dept_name
FROM
  Course;
```

CourseID	dept_name
201	CSE
202	ECE
211	ME
213	SIGCOM
222	EE
301	CY
303	ME
323	IT
506	MS
511	PH
518	CH
523	BIO
604	CY
655	CSE
702	BIO
710	VLSI
716	SIGCOM
722	CSE

*Note: If you want to retrieve all columns, use * as shown above.*

From Clause

It is a SQL clause that specifies the tables from which to retrieve data for a query.

Syntax:

```
SELECT column1, column2, ...
FROM table_name;
```

Examples:

```
SELECT
  *
FROM
  Course;
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
202	ASE	ECE	3
211	FD	ME	3
213	DSP	SIGCOM	3
222	BCS	EE	4
301	RC	CY	4
303	MET	ME	3
323	SE	IT	4
506	DM	MS	4
511	OPT	PH	3
518	ORG	CH	4
523	NEO	BIO	4
604	WLD	CY	3
655	CNS	CSE	3
702	MIB	BIO	4
710	ECM	VLSI	3
716	SPT	SIGCOM	4
722	TBS	CSE	4

*Note: If you want to retrieve all columns, use * as shown above.*

Where Clause

The "WHERE" clause in SQL is used to filter data from a table based on a specified set of conditions.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Examples:

```
SELECT
  *
FROM
  Course
WHERE
  dept_name = 'CSE';
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
655	CNS	CSE	3
722	TBS	CSE	4

ORDER BY

ORDER BY is used to sort the result set by one or more columns, in ascending or descending order.

Syntax:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name [ASC|DESC]
```

Examples:

```
SELECT
  *
FROM
  Course
ORDER BY
  credits DESC;
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
722	TBS	CSE	4
716	SPT	SIGCOM	4
222	BCS	EE	4
301	RC	CY	4
702	MIB	BIO	4
323	SE	IT	4
506	DM	MS	4
518	ORG	CH	4
523	NEO	BIO	4
511	OPT	PH	3
710	ECM	VLSI	3
655	CNS	CSE	3
604	WLD	CY	3
303	MET	ME	3
213	DSP	SIGCOM	3
211	FD	ME	3
202	ASE	ECE	3

Distinct

DISTINCT is used in a SELECT statement to return only unique values from a column.

Syntax:

```
SELECT DISTINCT
  (dept_name)
FROM
  Course;
```

Examples:

```
SELECT DISTINCT
  (dept_name)
FROM
  Course;
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
202	ASE	ECE	3
211	FD	ME	3
213	DSP	SIGCOM	3
222	BCS	EE	4
301	RC	CY	4
303	MET	ME	3
323	SE	IT	4
506	DM	MS	4
511	OPT	PH	3
518	ORG	CH	4
523	NEO	BIO	4
604	WLD	CY	3
655	CNS	CSE	3
702	MIB	BIO	4
710	ECM	VLSI	3
716	SPT	SIGCOM	4
722	TBS	CSE	4

The Table

Choose the dept_name column

dept_name
CSE
ECE
ME
SIGCOM
EE
CY
ME
IT
MS
PH
CH
BIO
CY
CSE
BIO
VLSI
SIGCOM
CSE

Choose the distinct values only

distinct(dept_name)
BIO
CH
CSE
CY
ECE
EE
IT
ME
MS
PH
SIGCOM
VLSI

Output

Aliases

In SQL, aliases are used to give a table, or a column in a table, a temporary name.

Syntax:

```
SELECT DISTINCT
  (dept_name) AS distinct_dept_name
FROM
  Course;
```

Examples:

```
SELECT DISTINCT
  (dept_name) AS distinct_dept_name
FROM
  Course;
```

distinct_dept_name
BIO
CH
CSE
CY
ECE
EE
IT
ME
MS
PH
SIGCOM
VLSI

Operators

NOT EQUAL TO

<> or ! =

Examples:

```
SELECT
  *
FROM
  Course
WHERE
  dept_name <> 'CSE';
```

CourseID	title	dept_name	credits
202	ASE	ECE	3
211	FD	ME	3
213	DSP	SIGCOM	3
222	BCS	EE	4
301	RC	CY	4
303	MET	ME	3
323	SE	IT	4
506	DM	MS	4
511	OPT	PH	3
518	ORG	CH	4
523	NEO	BIO	4
604	WLD	CY	3
702	MIB	BIO	4
710	ECM	VLSI	3
716	SPT	SIGCOM	4

BETWEEN Operator

The BETWEEN operator is used in SQL to check whether a value is within a specified range of values.

Examples:

```
SELECT
  *
FROM
  Course
WHERE
  credits BETWEEN 4 AND 5;
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
222	BCS	EE	4
301	RC	CY	4
323	SE	IT	4
506	DM	MS	4
518	ORG	CH	4
523	NEO	BIO	4
702	MIB	BIO	4
716	SPT	SIGCOM	4
722	TBS	CSE	4

LIKE Operator

The LIKE operator in SQL is used to perform pattern matching on string values.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

Pattern:

```
%: Matches any string of zero or more characters.
_: Matches any single character.
```

Examples:

Select titles starting with a D.

```
SELECT
  CourseID,
  title
FROM
  Course
WHERE
  title LIKE "D%";
```

CouseID	title
201	DSA
213	DSP
506	DM

Select titles ending with a D.

```
SELECT
  CourseID,
  title
FROM
  Course
WHERE
  title LIKE "%D";
```

CouseID	title
211	FD
604	WLD

Select dept_name containing the substring with a "GCO".

```
SELECT
  CourseID,
  dept_name
FROM
  Course
WHERE
  dept_name LIKE "%GCO%";
```

CouseID	title
213	SIGCOM
716	SIGCOM

*There are other various operators in SQL such as : Logical Operators: AND, OR, NOT , Mathematical Operators: +, -, *, /, %, String Operators: || (Concatenation), Comparison Operators: =, <>, <, >, <=, >=*

Grouping Clauses

GROUP BY Clause

The GROUP BY clause is used to group rows that have the same values.

Syntax:

```
SELECT column_name(s)
FROM table_name
GROUP BY column_name(s);
```

Examples:

```
SELECT
  dept_name,
  SUM(credits)
FROM
  Course
GROUP BY
  dept_name;
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
202	ASE	ECE	3
211	FD	ME	3
213	DSP	SIGCOM	3
222	BCS	EE	4
301	RC	CY	4
303	MET	ME	3
323	SE	IT	4
506	DM	MS	4
511	OPT	PH	3
518	ORG	CH	4
523	NEO	BIO	4
604	WLD	CY	3
655	CNS	CSE	3
702	MIB	BIO	4
710	ECM	VLSI	3
716	SPT	SIGCOM	4
722	TBS	CSE	4

Same value of dept_name i.e. CSE, thus group these rows and sum the values of 'credits' column.

5 + 3 + 4 = 12

Do the same with all distinct values of dept_name

OUTPUT

dept_name	Sum(credits)
BIO	8
CH	4
CSE	12
CY	7
ECE	3
EE	4
IT	4
ME	6
MS	4
PH	3
SIGCOM	7
VLSI	3

HAVING Clause

The HAVING clause is used to filter groups based on a condition that aggregates cannot handle.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition;
```

Examples:

```
SELECT
  dept_name,
  SUM(credits)
FROM
  Course
GROUP BY
  dept_name
HAVING
  SUM(credits) > 5;
```

CourseID	title	dept_name	credits
201	DSA	CSE	5
202	ASE	ECE	3
211	FD	ME	3
213	DSP	SIGCOM	3
222	BCS	EE	4
301	RC	CY	4
303	MET	ME	3
323	SE	IT	4
506	DM	MS	4
511	OPT	PH	3
518	ORG	CH	4
523	NEO	BIO	4
604	WLD	CY	3
655	CNS	CSE	3
702	MIB	BIO	4
710	ECM	VLSI	3
716	SPT	SIGCOM	4
722	TBS	CSE	4

Same value of dept_name i.e. CSE, thus group these rows and sum the values of 'credits' column.

5 + 3 + 4 = 12

Do the same with all distinct values of dept_name

OUTPUT

dept_name	Sum(credits)
BIO	8
CH	4
CSE	12
CY	7
ECE	3
EE	4
IT	4
ME	6
MS	4
PH	3
SIGCOM	7
VLSI	3

Apply the Filter Condition

OUTPUT

dept_name	Sum(credits)
BIO	8
CSE	12
CY	7
ME	6
SIGCOM	7

Aggregate Functions

Count

The COUNT() function returns the number of rows that match a specified condition or expression.

Syntax:

```
COUNT([Distinct] Expression)
```

Examples:

```
SELECT
  COUNT(*)
FROM
  Course;
```

COUNT(*)
18

```
SELECT
  COUNT(DISTINCT dept_name)
FROM
  Course;
```

COUNT(Distinct dept_name)
12

Sum

The SUM() function in SQL returns the sum of all the values in a selected column.

Syntax:

```
SUM(expression)
```

Examples:

```
SELECT
  SUM(credits) AS total_cse_credits
FROM
  Course
WHERE
  dept_name = 'CSE';
```

total_cse_credits
12

Max

The MAX() function in SQL returns the maximum value of a selected column.

Syntax:

```
MAX(expression)
```

Examples:

```
SELECT
  MAX(credits) AS maximum_credits
FROM
  Course
WHERE
  dept_name = 'CSE';
```

maximum_credits
5

Min

The MIN() function in SQL returns the minimum value of a selected column.

Syntax:

```
MIN(expression)
```

Examples:

```
SELECT
  MIN(credits) AS minimum_credits
FROM
  Course
WHERE
  dept_name = 'CSE';
```

minimum_credits
3

Avg

The AVG() function in SQL returns the average value of a selected column.

Syntax:

```
AVG(expression)
```

Examples:

```
SELECT
  AVG(credits)
FROM
  Course
WHERE
  dept_name = 'CSE';
```

Avg(credits)
4.0000

Join in SQL

INNER JOIN

JOIN (or explicitly INNER JOIN) returns rows that have matching values in both tables.

Syntax:

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

LEFT JOIN

LEFT JOIN returns all rows from the left table with corresponding rows from the right table.
If there's no matching row, NULLs are returned as values from the second table.

Syntax:

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

RIGHT JOIN

RIGHT JOIN returns all rows from the right table with corresponding rows from the left table.
If there's no matching row, NULLs are returned as values from the left table.

Syntax:

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

FULL JOIN

FULL JOIN (or explicitly FULL OUTER JOIN) returns all rows from both tables.
If there's no matching row in the second table, NULLs are returned.

Syntax:

```
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

SELF JOIN

A self join is a regular join, but the table is joined with itself.
A self join is used to join a table to itself.

Syntax:

```
SELECT column_name(s)
FROM table_name1
SELF JOIN table_name1;
```

OR

```
SELECT column(s)
FROM table_name1, table_name1;
```

CROSS JOIN

CROSS JOIN returns all possible combinations of rows from both tables.
There are two syntaxes available.

Syntax:

```
SELECT column(s)
FROM table_name1
CROSS JOIN table_name2;
```

OR

```
SELECT column(s)
FROM table_name1, table_name2;
```