coursera

SQL Basics Cheat Sheet

Creating a Database

Create Database:

- Creates a new database.

```
Unset
CREATE DATABASE my_database;
```

Use:

- Selects a database to use.

```
Unset
USE my_database;
```

Alter Database:

- Modifies an existing database.

```
Unset
ALTER DATABASE my_database
MODIFY NAME = new_database_name;
```

Drop Database:

- Deletes an existing database.

```
Unset
DROP DATABASE my_database;
```

Creating Data

Create Table:

- Creates a new table.

```
Unset
CREATE TABLE employees (
    id INT,
    name VARCHAR(50),
    age INT
);
```

Create Index:

- Creates an index for faster query performance.

```
Unset
CREATE INDEX idx_employee_name
ON employees (name);
```

Insert into:

- Inserts new rows into a table.

```
Unset
INSERT INTO employees (id, name, age)
VALUES (1, 'Alice', 30);
```

Alter Table:

- Modifies an existing table.

```
Unset
ALTER TABLE employees
ADD COLUMN department VARCHAR(50);
```

Drop Table:

- Deletes an entire table.

```
Unset
DROP TABLE employees;
```

Reading & Querying Data

Select:

- Retrieves data from a table.

```
Unset
SELECT * FROM employees;
```

Distinct:

- Retrieves unique values from a column.

```
Unset
SELECT DISTINCT department FROM employees;
```

Limit:

- Limits the number of rows returned by a query.

```
Unset
SELECT * FROM employees
LIMIT 5;
```

Offset:

- Specifies an offset for the rows returned by a query.

```
Unset
SELECT * FROM employees
LIMIT 5 OFFSET 10;
```

Fetch:

- Retrieves a specific number of rows.

```
Unset
SELECT * FROM employees
FETCH FIRST 5 ROWS ONLY;
```

Case:

- Provides conditional logic in a query.

```
Unset
SELECT name,
  CASE
    WHEN age < 30 THEN 'Young'
    ELSE 'Experienced'
  END as experience
FROM employees;
```

Updating & Manipulating Data

Update:

- Modifies existing rows in a table.

```
Unset
UPDATE employees
SET age = 31
WHERE id = 1;
```

Column constraints:

- Sets rules for column values.

```
Unset
ALTER TABLE employees
ADD CONSTRAINT unique_name UNIQUE (name);
```

Primary key:

- Uniquely identifies each row.

```
Unset
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(50)
);
```

Updating & Manipulating Data (continued)

Unique:

- Ensures all values in a column are unique.

```
Unset
ALTER TABLE employees
ADD CONSTRAINT unique_name UNIQUE (name);
```

Not null:

- Ensures a column cannot have NULL values.

```
Unset
ALTER TABLE employees
MODIFY COLUMN name VARCHAR(50) NOT NULL;
```

Default:

- Sets a default value for a column.

```
Unset
ALTER TABLE employees
ADD COLUMN hire_date DATE DEFAULT CURRE
```

Filtering Data

Where:

- Filters records based on a condition.

```
Unset
SELECT * FROM employees
WHERE age > 30;
```

Like:

- Filters records using pattern matching.

```
Unset
SELECT * FROM employees
WHERE name LIKE 'A%';
```

In:

- Filters records that match a list of values.

```
Unset
SELECT * FROM employees
WHERE department IN ('HR', 'IT');
```

Between:

- Filters records within a range.

```
Unset
SELECT * FROM employees
WHERE age BETWEEN 25 AND 35;
```

Is Null:

- Filters records with NULL values.

```
Unset
SELECT * FROM employees
WHERE department IS NULL;
```

Order by:

- Sorts records in ascending or descending order.

```
Unset
SELECT * FROM employees
ORDER BY name ASC;
```

SQL Operators

AND:

- Combines multiple conditions.

```
Unset
SELECT * FROM employees
WHERE age > 30 AND department = 'IT';
```

OR:

- At least one of the conditions must be true.

```
Unset
SELECT * FROM employees
WHERE age > 30 OR department = 'HR';
```

NOT:

- Excludes specified condition.

```
Unset
SELECT * FROM employees
WHERE NOT department = 'HR';
```

LIKE:

- Searches for a specified pattern.

```
Unset
SELECT * FROM employees
WHERE name LIKE 'A%';
```

IN:

- Matches any value in a list.

```
Unset
SELECT * FROM employees
WHERE department IN ('HR', 'Finance');
```

Between:

- Matches values within a range.

```
Unset
SELECT * FROM employees
WHERE age BETWEEN 25 AND 35;
```

IS NULL:

- Matches NULL values.

```
Unset
SELECT * FROM employees
WHERE department IS NULL;
```

ORDER BY:

- Sorts the result set.

```
Unset
SELECT * FROM employees
ORDER BY age DESC;
```

GROUP BY:

- Groups rows sharing a property.

```
Unset
SELECT department, COUNT(*)
FROM employees
GROUP BY department;
```

Aggregate Data

COUNT:

- Counts the number of rows.

```
Unset
SELECT COUNT(*) FROM employees;
```

SUM:

- Calculates the sum of a column.

```
Unset
SELECT SUM(salary) FROM employees;
```

AVG:

- Calculates the average value.

```
Unset
SELECT AVG(age) FROM employees;
```

MIN:

- Finds the minimum value.

```
Unset
SELECT MIN(age) FROM employees;
```

MAX:

- Finds the maximum value.

```
Unset
SELECT MAX(age) FROM employees;
```

GROUP BY:

- Groups rows that have the same values.

```
Unset
SELECT department, COUNT(*)
FROM employees
GROUP BY department;
```

HAVING:

- Filters groups based on a condition.

```
Unset
SELECT department, COUNT(*)
FROM employees
GROUP BY department
HAVING COUNT(*) > 5;
```

Constraints

PRIMARY KEY:

- Uniquely identifies each row in a table.

```
Unset
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(50)
);
```

FOREIGN KEY:

- Uniquely identifies a row in another table.

```
Unset
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    employee_id INT,
    FOREIGN KEY (employee_id) REFERENCES employees(id)
);
```

UNIQUE:

- Ensures all values in a column are unique.

```
Unset
ALTER TABLE employees
ADD CONSTRAINT unique_name UNIQUE (name);
```

Multiple Tables

OUTER JOIN:

- Returns rows when there is a match in one of the tables.

```
Unset
SELECT employees.name, orders.order_id
FROM employees
LEFT JOIN orders ON employees.id = orders.employee_id;
```

WITH:

- Creates a named temporary result set.

```
Unset
WITH department_count AS (
    SELECT department, COUNT(*) AS num
    FROM employees
    GROUP BY department
)
SELECT * FROM department_count;
```

UNION:

- Combines the result sets of two queries.

```
Unset
SELECT name FROM employees
UNION
SELECT name FROM managers;
```

CROSS JOIN:

- Returns the Cartesian product of both tables.

```
Unset
SELECT employees.name, departments.name
FROM employees
CROSS JOIN departments;
```

INNER JOIN:

- Returns rows with a match in both tables.

```
Unset
SELECT employees.name, departments.name
FROM employees
INNER JOIN departments ON employees.department
```

SQL Functions

Aggregate Functions:

- **SELECT AVG:** Calculates average value.

```
Unset
SELECT AVG(salary) FROM employees;
```

String Functions:

- **SELECT CONCAT:** Concatenates two or more strings.

```
Unset
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM
employees;
```

- **SELECT SUBSTR:** Extracts a substring from a string.

```
Unset
SELECT SUBSTR(name, 1, 3) AS short_name FROM employees;
```

- **SELECT INSERT:** Inserts a substring into a string.

```
Unset
SELECT INSERT(name, 1, 0, 'Dr. ') AS titled_name FROM
employees;
```

- **SELECT CURRENT_DATE:** Retrieves the current date.

```
Unset
SELECT CURRENT_DATE;
```

- **SQRT():** Calculates the square root of a number.

```
Unset
SELECT SQRT(salary) FROM employees
```