Should Know for DATA analyse

1. **Aggregate Functions:** These functions operate on groups of data to compute summarized values, such as averages, sums, and counts. Examples include:

   a. AVG(column_name): Calculates the average of a set of values.

   b. SUM(column_name): Computes the sum of a set of values.

   c. COUNT(column_name): Determines the number of non-NULL values in a column.

2. **Grouping and Aggregation:** The GROUP BY clause allows you to group data based on specific criteria and apply aggregate functions to each group. For instance:

   SQL

   ```
   SELECT category, AVG(price) FROM menu GROUP BY category;
   ```

   Use code with caution. Learn morecontent_copy

   This query groups menu items by category and calculates the average price for each category.

3. **Conditional Expressions:** The CASE statement enables you to perform conditional logic within queries, allowing for more nuanced data analysis. For example:

   SQL

   ```
   SELECT order_id,
          CASE
            WHEN status = 'pending' THEN 'Order in Progress'
            WHEN status = 'preparing' THEN 'Order Being Prepare
            WHEN status = 'delivered' THEN 'Order Delivered'
   ```

```
        END AS order_status
FROM orders;
```

Use code with caution. Learn morecontent_copy

This query classifies orders based on their status and assigns a descriptive label to each status.

4. **Window Functions:** SQLite3 provides window functions that perform calculations over sliding windows within a result set. Examples include:

   a. ROW_NUMBER(): Assigns a sequential row number to each row within a partitioned result set.

   b. LAG(): Retrieves the value from a specified number of rows earlier.

   c. LEAD(): Retrieves the value from a specified number of rows later.

5. **String Manipulation Functions:** SQLite3 offers functions to manipulate text data, such as:

   a. SUBSTR(text, start, length): Extracts a substring from a text string.

   b. UPPER(text): Converts a text string to uppercase.

   c. LOWER(text): Converts a text string to lowercase.

   d. TRIM(text): Removes leading and trailing whitespace from a text string.

These functions are powerful tools for analyzing and transforming data within SQLite3, enabling you to gain valuable insights from your data.

@swapnokpanda

SQL CHEATSHEET

✔ Genuine
✔ Authentic
✔ Quality

Categories

DDL : Data Definition Language
DQL : Data Query Language
DML : Data Manipulation Language
DCL : Data Control Language
TCL : Transaction Control Language

Commands

DDL
CREATE | DROP | ALTER | TRUNCATE
RENAME | COMMENT

DQL
SELECT

DML
INSERT | UPDATE | DELETE | LOCK
CALL | EXPLAIN PLAN

DCL
GRANT | REVOKE

TCL
COMMIT | ROLLBACK
SAVEPOINT | SET TRANSACTION

Operators

Arithmetic
+ - * / %

Bitwise
& | ^

Comparison
= < > <= >= !< !> <> !=

Compound
+= -= *= /= %= &= |= ^=

Logical
AND | OR | NOT | ANY
SOME | ALL | BETWEEN
IN | EXISTS | LIKE
IS NULL | UNIQUE

Important Keywords

WHERE | DISTINCT | LIMIT
ORDER BY | DESC | ASC
AS | FROM | SET | VALUES
CASE | DEFAULT

Database Objects

TABLE | VIEW | SYNONYM
SEQUENCE | INDEX | TRIGGER

Constraints

NOT NULL | UNIQUE
PRIMARY KEY | FOREIGN KEY
CHECK | DEFAULT

Aggregation Functions

AVG | COUNT
MAX | MIN | SUM

Aggregation Keywords

GROUP BY | HAVING

Joins

INNER JOIN

LEFT [OUTER] JOIN

FULL [OUTER] JOIN

RIGHT [OUTER] JOIN

Set Operations

UNION
UNION ALL

INTERSECT

EXCEPT
MINUS

DDL Examples

Create a Table
```
CREATE TABLE Students(
    rollno int PRIMARY KEY,
    fname varchar(255) NOT NULL,
    lname varchar(255)
);
```

Adding a new column to the Table
```
ALTER TABLE Students
ADD email varchar(255);
```

Modifying the data type of existing column
```
ALTER TABLE Students
ALTER COLUMN lname varchar(512);
```

Removing an existing column from the Table
```
ALTER TABLE Students
DROP COLUMN email;
```

Truncate (remove all data) a Table
```
TRUNCATE TABLE Students;
```

Drop a Table
```
DROP TABLE Students;
```

DQL Examples

Fetch all data from a Table
```
SELECT * FROM Students;
```

Filter data from a Table
```
SELECT * FROM Students
WHERE rollno = 1234;

SELECT * FROM Students
WHERE rollno > 1234
AND   age < 15;
```

Fetch selected columns
```
SELECT fname, lname
FROM   Students
WHERE rollno > 1234
AND   age < 15;
```

Fetch maximum 10 rows
```
SELECT fname, lname
FROM   Students
WHERE rollno > 1234
AND   age < 15
LIMIT  10;
```

Fetch count of records
```
SELECT count(*)
FROM   Students;
```

Fetch Maximum Age
```
SELECT max(age)
FROM   Students;
```

Fetch Minimum Age
```
SELECT min (age)
FROM   Students;
```

Fetch Sum of Age
```
SELECT sum(age)
FROM   Students;
```

Fetch Average Age
```
SELECT avg (age)
FROM   Students;
```

Fetch Average Age for each gender
```
SELECT avg (age)
FROM   Students
GROUP BY gender;
```

Sort (order) fetched records
```
SELECT fname, lname
FROM   Students
WHERE rollno > 1234
AND   age < 15
ORDER BY gender;
```

Sort in descending order
```
SELECT fname, lname
FROM   Students
WHERE rollno > 1234
AND   age < 15
ORDER BY gender DESC;
```

Fetch from 2 Tables
```
SELECT fname, clsteacher
FROM   Students
INNER JOIN Section
ON     Students.section
       = Section.id;
```

DML Examples

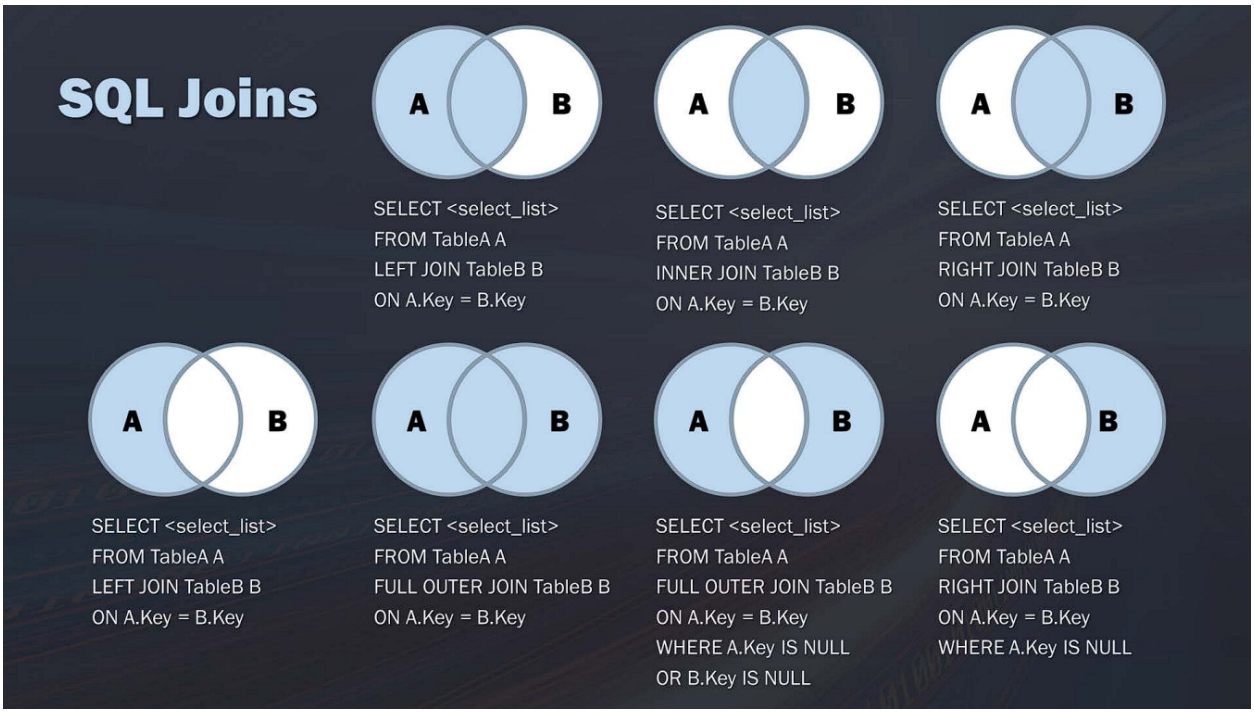
Insert data (rows) into a Table
```
INSERT INTO Students (rollno, fname, lname)
VALUES (1234, 'Christiano', 'Ronaldo');
```

Update data (value of column) of a Table
```
UPDATE Students SET lname = 'Messi'
WHERE rollno = 1234;
```

Delete data (rows) from a Table
```
DELETE FROM Students WHERE rollno = 1234;
```

Aggregate and, Filter
```
SELECT section, count (*) AS studentcount
FROM   Students
GROUP BY section
HAVING count (*) > 20;
```

Full Outer Join
```
SELECT fname, clsteacher
FROM   Students
FULL JOIN Section
ON     Students.section = Section.id;
```


SQL Joins
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL