#_ The SQL Handbook: +100 SQL Concepts

1. Basic SQL Concepts:

1- SQL:

Stands **for** "Structured Query Language". It's a standard language **for** managing **and** manipulating databases.

2- Relational Databases:

These are databases structured to recognize relations **among** stored **items of** information. Example: MySQL, PostgreSQL, Oracle DB.

3- Tables:

In SQL, **a** table is **a** collection **of** related data held **in a** structured format **within a** database. It consists **of** columns **and** rows.

4- Columns:

These are set of data values of a particular simple type, one value **for** each row of the database.

5- Rows:

Also called a record **or** tuple, a row **is** a **set of** data values **that** are interrelated.

6- Data Types:

Each column **in a** SQL **table** has **a** related data type. SQL has several data types like INT, VARCHAR, DATE, BOOLEAN, etc.

7- SELECT:

The SELECT statement is used to select data from a database.

```
SELECT column_name FROM table_name;
```

By: Waleed Mousa

8- **FROM**:

FROM clause **is** used **to** specify the table **to select or** delete data **from**.

9- WHERE:

The WHERE clause is used to filter records.

```
SELECT column_name FROM table_name WHERE condition;
```

10- INSERT INTO:

The INSERT INTO statement is used to insert new records in a table.

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

11- UPDATE:

The UPDATE statement is used to modify the existing records in a table.

```
UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;
```

12- DELETE:

The DELETE statement is used to delete existing records in a table.

```
DELETE FROM table_name WHERE condition;
```

13- Operators:

SQL uses operators **like** '=', '<>', '>', '<', '>=', '<=', 'BETWEEN', 'LIKE', 'IN' etc.

14- AND, OR, NOT:

The AND, OR and NOT operators are used to filter records based on more than one condition.

```
SELECT column1, column2 FROM table_name WHERE condition1 AND condition2;
```

By: Waleed Mousa

15- ORDER BY:

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

```
SELECT column_name FROM table_name ORDER BY column_name ASC|DESC;
```

16- DISTINCT:

The DISTINCT keyword is used to return only distinct (different) values.

```
SELECT DISTINCT column_name FROM table_name;
```

17- COUNT, AVG, SUM:

The COUNT(), AVG() and SUM() functions return a count, average and sum of the numeric column values.

```
SELECT COUNT(column_name) FROM table_name;
SELECT AVG(column_name) FROM table_name;
SELECT SUM(column_name) FROM table_name;
```

18- GROUP BY:

The GROUP BY statement groups rows that have the same values in specified columns into aggregated data.

```
SELECT column_name, COUNT(column_name) FROM table_name GROUP BY column_name;
```

19- HAVING:

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
SELECT column_name, COUNT(column_name) FROM table_name GROUP BY column_name
HAVING COUNT(column_name) > value;
```

20- CREATE DATABASE:

The CREATE DATABASE statement is used to create a new SQL database.

```
CREATE DATABASE database_name;
```

By: Waleed Mousa

21- DROP DATABASE:

The DROP DATABASE statement is used to drop an existing SQL database.

```
DROP DATABASE database_name;
```

22- CREATE TABLE:

The CREATE TABLE statement is used to create a new table in a database.

```
CREATE TABLE table_name (column1 datatype, column2 datatype, column3
datatype);
```

23- DROP TABLE:

The DROP TABLE statement is used to drop an existing table in a SQL database.

```
DROP TABLE table_name;
```

24- ALTER TABLE:

The ALTER TABLE statement is used to add, delete/drop or modify columns in an existing table.

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

25- Constraints:

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. Constraints can be column level or table level.

26- PRIMARY KEY:

A PRIMARY KEY is a constraint that uniquely identifies each record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.

```
CREATE TABLE table_name (column1 datatype PRIMARY KEY, column2 datatype,
column3 datatype);
```

By: Waleed Mousa

27- FOREIGN KEY:

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

28- CHECK:

The CHECK constraint is used to limit the value range that can be placed in a column.

```
CREATE TABLE table_name (
    column1 datatype CONSTRAINT chk_column CHECK (condition),
    column2 datatype, column3 datatype
);
```

29- UNIQUE:

The UNIQUE constraint ensures that all values in a column are different.

```
CREATE TABLE table_name (
    column1 datatype UNIQUE,
    column2 datatype, column3 datatype
);
```

30- INDEX:

Indexes are used to retrieve data from the database more quickly than otherwise.

```
CREATE INDEX index_name ON table_name (column1, column2);
```

31- AUTO INCREMENT:

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

```
CREATE TABLE table_name (
    ID int NOT NULL AUTO_INCREMENT,
    column1 datatype, column2 datatype,
    PRIMARY KEY (ID)
);
```

32- DATE:

SQL uses the DATE data type to store date data.

```
SELECT column_name FROM table_name WHERE DATE(column_name) = 'YYYY-MM-DD';
```

33- NULL:

> The NULL value represents **a** missing unknown data.

34- IS NULL/IS NOT NULL:

IS NULL and IS NOT NULL are operators used with the WHERE clause to test for empty values.

```
SELECT column_name FROM table_name WHERE column_name IS NULL;
SELECT column_name FROM table_name WHERE column_name IS NOT NULL;
```

35- LIKE:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

```
SELECT column_name FROM table_name WHERE column_name LIKE pattern;
```

36- IN:

The IN operator allows you to specify multiple values in a WHERE clause.

```
SELECT column_name FROM table_name WHERE column_name IN (value1, value2);
```

37- BETWEEN:

The BETWEEN operator selects values within a given range inclusive.

```
SELECT column_name FROM table_name WHERE column_name BETWEEN value1 AND
value2;
```

38- JOIN:

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

39- INNER JOIN:

The INNER JOIN keyword selects records that have matching values in both tables.

```
SELECT employees.name, departments.dept_name FROM employees INNER JOIN
departments ON employees.id = departments.employee_id;
```

40- LEFT (OUTER) JOIN:

The LEFT JOIN keyword returns all records from the left table, and the matched records from the right table.

```
SELECT employees.name, departments.dept_name FROM employees LEFT JOIN
departments ON employees.id = departments.employee_id;
```

41- RIGHT (OUTER) JOIN:

The RIGHT JOIN keyword returns all records from the right table, and the matched records from the left table.

```
SELECT employees.name, departments.dept_name FROM employees RIGHT JOIN
departments ON employees.id = departments.employee_id;
```

42- FULL (OUTER) JOIN:

The FULL OUTER JOIN keyword returns all records when there is a match in either left or right table records.

```
SELECT employees.name, departments.dept_name FROM employees FULL JOIN
departments ON employees.id = departments.employee_id;
```

43- UNION:

The UNION operator is used to combine the result-set of two or more SELECT statements. Each SELECT statement within UNION must have the same number of columns.

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

44- UNION ALL:

UNION ALL is used to combine the result-set of two or more SELECT statements with duplicate values.

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

45- SELECT INTO:

The SELECT INTO statement copies data from one table into a new table.

```
SELECT column_name(s)
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

46- INSERT INTO SELECT:

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

By: Waleed Mousa

47- CASE:

The CASE statement goes through conditions and returns a value when the first condition is met.

```
SELECT column_name,
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    ELSE result3
END
FROM table_name;
```

48- Stored Procedure:

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

49- TRIGGER:

A trigger is a stored procedure in a database that automatically reacts to an event like insertions, updates, or deletions in a specific table.

```
CREATE TRIGGER trigger_name
ON table_name
AFTER INSERT/UPDATE/DELETE
AS
sql_statement;
```

50- Views:

In SQL, a view is a virtual table based on the result-set of an SQL statement.

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition;
```

By: Waleed Mousa

2. Intermediate SQL Concepts:

1- Aliases:

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

```
SELECT column_name AS alias_name FROM table_name;
```

2- Self-JOIN:

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

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

3- GROUP_CONCAT():

This function concatenates strings from a group into a single string with various options.

```
SELECT GROUP_CONCAT(column_name SEPARATOR ', ') FROM table_name;
```

4- Handling Duplicates:

```
SQL provides several ways to handle duplicate records in a table - IGNORE,
REPLACE, and ON DUPLICATE KEY UPDATE.
```

4.1- *The following example uses the IGNORE keyword to prevent the insertion of duplicate rows:*

```
INSERT IGNORE INTO table_name (column1, column2) VALUES ('value1', 'value2');
```

4.2- *For the REPLACE keyword, if a duplicate is found, the old row is deleted before the new row is inserted:*

```
REPLACE INTO table_name (column1, column2) VALUES ('value1', 'value2');
```

4.3- *The ON DUPLICATE KEY UPDATE statement, on the other hand, updates the row if a duplicate is found:*

```
INSERT INTO table_name (column1, column2) VALUES ('value1', 'value2')
ON DUPLICATE KEY UPDATE column1 = 'value1', column2 = 'value2';
```

By: Waleed Mousa

5- Transactions:

Transactions group a set of tasks into a single execution unit.

```
START TRANSACTION;
INSERT INTO table1 (column1) VALUES ('value1');
UPDATE table2 SET column2 = 'value2' WHERE column1 = 'value1';
DELETE FROM table3 WHERE column1 < 100;
COMMIT;
```

6- Locking:

SQL uses locks to control concurrent access to data.

```
START TRANSACTION;
SELECT * FROM table_name WHERE column1 = 'value1' FOR UPDATE;
```

7- Indexing:

Indexing is used to speed up the retrieval of records on a database table.

```
CREATE INDEX index_name
ON table_name (column_name);
```

8- Normalization:

```
Normalization is a database design technique which organizes tables in a
manner that reduces redundancy and dependency of data.
```

9- Denormalization:

```
Denormalization is a strategy used on a previously-normalized database to
increase performance.
```

10- Subquery:

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

```
SELECT column_name(s) FROM table_name1 WHERE column_name operator (SELECT
column_name(s) from table_name2);
```

By: Waleed Mousa

11- Join Optimization:

> Consider you have two tables, table1 **and** table2, **and** you are performing a **JOIN** operation. Depending **on** the DBMS **and** the indexes used, you might have different types **of** joins available, such **as** hash **join or** merge **join**. Generally, DBMS takes care **of** choosing the optimal **join**, but sometimes you may need **to** manually hint the **join type**.

12- Cursor Manipulation:

SQL cursors are database objects used to manipulate rows from a result set on a row-by-row basis.

```
DECLARE @MyCursor CURSOR;
DECLARE @MyField VARCHAR(50);

BEGIN
     SET @MyCursor = CURSOR FOR
     SELECT MyField FROM MyTable WHERE MyCondition;

     OPEN @MyCursor
     FETCH NEXT FROM @MyCursor
     INTO @MyField;

     WHILE @@FETCH_STATUS = 0
     BEGIN
     -- Here is where you process each row.

     FETCH NEXT FROM @MyCursor
     INTO @MyField;
     END;

     CLOSE @MyCursor ;
     DEALLOCATE @MyCursor;
END;
```

13- Exception Handling:

> Handling SQL exceptions correctly **is** essential **in** building a robust database application.

By: Waleed Mousa

14- Array Manipulation:

Arrays can be used **in** SQL to hold more than one value **at a** time.

15- JSON Data:

SQL Server 2016 and other modern databases provide support for storing, querying, and indexing JSON data.

```
SELECT
     json_extract(json_column, '$.key') as extracted_key
FROM
     json_table;
```

16- XML Data:

Many databases provide support for XML data and can query it using XQuery.

```
SELECT
     XQuery('for $t in //row return string($t/column_name)')
FROM
     xml_table;
```

17- Regular Expressions:

Some databases support querying **data** using regular expressions.

18- Full-Text Search:

Full-text search is a technique for searching a computer-stored document or database.

```
SELECT
     column
FROM
     table
WHERE
     MATCH(column) AGAINST ('keyword');
```

By: Waleed Mousa

19- Binary Data Handling:

> **Binary** large objects (**BLOBs**) can **be** stored in a SQL database.

21- Database Views:

A view is a virtual table based on the result-set of an SQL statement.

```sql
CREATE VIEW view_name AS
SELECT column1, column2
FROM table_name
WHERE condition;
```

22- Database Collation:

> Collation is a **set of rules** that tell **database engine** how **to** compare **and sort** the character **data in SQL** Server.

By: Waleed Mousa

3. Advanced SQL Concepts:

1- Correlated Subqueries:

A correlated subquery, however, depends on the outer query. It's a subquery that uses values from the outer query.

```
SELECT column_name(s) FROM table_name1 outer WHERE column_name operator
(SELECT column_name(s) FROM table_name2 inner WHERE outer.column_name =
inner.column_name);
```

2- EXISTS:

The EXISTS operator is used to test for the existence of any record in a subquery.

```
SELECT column_name(s) FROM table_name1 WHERE EXISTS (SELECT column_name FROM
table_name2 WHERE condition);
```

3- Common Table Expressions (CTEs):

A CTE provides the significant advantage of being able to reference itself, thereby creating a recursive CTE.

```
WITH RECURSIVE cte_name (column_name(s)) AS (
    SQL query
)

SELECT * FROM cte_name;
```

4- PIVOT:

> Pivoting data can be achieved **in** SQL **by using** aggregate functions **in** concert **with a** CASE statement **in the** query.

5- Window Functions:

> Window functions provide **the** ability to perform calculations across sets **of** rows that are related to **the** current query row.

By: Waleed Mousa

6- RANK():

The RANK() function is a window function that assigns a unique rank to each row within the partition of a result set.

```
SELECT column_name(s), RANK() OVER (ORDER BY column_name) FROM table_name;
```

7- DENSE_RANK():

This function provides the same functionality as RANK(), but in the event of a tie, it doesn't skip any ranks.

```
SELECT column_name(s), DENSE_RANK() OVER (ORDER BY column_name) FROM
table_name;
```

8- ROW_NUMBER():

This function assigns a unique row number for each row, but makes no promise about what order that will be in, or even that the order will be deterministic.

```
SELECT column_name(s), ROW_NUMBER() OVER (ORDER BY column_name) FROM
table_name;
```

9- NTILE():

This function distributes the rows in an ordered partition into a specified number of groups, or buckets, and assigns a unique bucket number to each row in the partition.

```
SELECT column_name(s), NTILE(bucket_number) OVER (ORDER BY column_name) FROM
table_name;
```

10- LAG() and LEAD():

These functions fetch the value of a given expression for the previous row (LAG) or the next row (LEAD) in the same result set without the use of a self-join.

```
SELECT column_name, LAG(column_name) OVER (ORDER BY column_name),
LEAD(column_name) OVER (ORDER BY column_name) FROM table_name;
```

By: Waleed Mousa

11- FIRST_VALUE() and LAST_VALUE():

These functions return the first or the last value from an ordered set of values in SQL.

```
SELECT column_name, FIRST_VALUE(column_name) OVER (ORDER BY column_name),
LAST_VALUE(column_name) OVER (ORDER BY column_name) FROM table_name;
```

12- CUME_DIST():

This function computes the cumulative distribution of a value in a group of values in SQL. That is, CUME_DIST computes the relative position of a specified value in a group of values.

```
SELECT column_name, CUME_DIST() OVER (ORDER BY column_name) FROM table_name;
```

13- PERCENT_RANK():

This function computes the relative rank of a row returned by a query in SQL.

```
SELECT column_name, PERCENT_RANK() OVER (ORDER BY column_name) FROM
table_name;
```

14- Database Administration:

This involves a wide array of operations, **from** managing users **and** permissions, **to** performance optimization, backups, **and** migrating data between systems.

15- Materialized Views:

These are similar to regular views, but the results are stored **in a** physical **table for** performance gains.

16- Analytic Functions:

These are a type of function that compute across a set of table rows that are somehow related **to** the current row.

By: Waleed Mousa

17- Sequences:

Sequences are database objects from which multiple users may generate unique integers.

18- Synonyms:

A synonym is an alias for a database object, providing a layer of abstraction that can simplify SQL statements for database users.

19- Partitioning:

This is a technique to divide a large database table into smaller, more manageable parts without having to create separate tables for each part.

20- User-Defined Functions (UDFs):

UDFs are functions defined by the user at the database level.

```
CREATE FUNCTION function_name (@param1 int, @param2 nvarchar(50))
RETURNS TABLE
AS
RETURN
(
      SELECT column1, column2
      FROM table_name
      WHERE column1 = @param1 AND column2 = @param2
);
```

21- Dynamic SQL:

Dynamic SQL allows programmers to write SQL statements that will be executed at runtime.

```
DECLARE @column_name VARCHAR(100);
SET @column_name = 'column1';
EXEC('SELECT ' + @column_name + ' FROM table_name');
```

By: Waleed Mousa

22- Recursive Queries:

Recursive queries are used to query hierarchical data.

```
WITH RECURSIVE recursive_query AS (
  SELECT column1, column2
  FROM table_name
  WHERE condition1
  UNION ALL
  SELECT r.column1, r.column2
  FROM table_name AS r
  JOIN recursive_query AS rq ON r.column3 = rq.column1
)
SELECT * FROM recursive_query;
```

23- Database Replication:

> Replication is a **set of** technologies **for** copying **and** distributing **data and database** objects **from** one **database to** another.

24- Database Sharding:

> Sharding is a type of database partitioning that separates large databases into smaller, faster, more easily managed parts.

25- Database Migration:

> The process **of** moving your data from one database engine to another.

26- Database Performance Tuning:

> A wide variety **of** practices used **to** make a database run faster.

27- Distributed Databases:

> A distributed database is a database that consists of two **or** more files located **in** different sites either on the same network **or** on entirely different networks.

By: Waleed Mousa

28- Database Security:

> Practices used **to** protect your database **from** intentional **or** accidental threats, risks, **or** attacks.

29- Database Backup and Restoration:

> Essential operations **for** preserving and recovering **data**.

30- Cloud Databases:

> Modern databases hosted **on the** cloud, providing benefits such **as** scalability **and** flexibility.

31- Data Lake and Data Lakehouse:

> A data lake is a storage repository that holds a vast amount of raw data. A data lakehouse blends the best elements of a data warehouse **and** a data lake.

By: Waleed Mousa

4. Performance Tuning Concepts:

1- Query Optimization:

This is the overall process of choosing the most efficient means of executing a SQL statement.

2- Execution Plan:

An execution plan is the sequence of operations that will be performed for a given query.

3- Indexes Optimization:

This is the process of choosing the right indexes for a database table to improve query performance.

4- Data Profiling:

This is the process of examining the data available from an existing information source and collecting statistics or informative summaries about it.

5- Data Modeling:

This refers to the practice of documenting a complex software system design as an easily understood diagram, using text and symbols to represent the way data needs to flow.

6- Database Design:

This is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language.

7- Data Partitioning:

This is the process of splitting up a large table across multiple storage locations in order to improve query performance.

8- Parallel Execution:

This is the process of carrying out multiple tasks or sequences of SQL operations simultaneously.

9- Scalability:

The capacity to handle increased workload by continually achieving higher throughput when resources are added.

10- Database Replication:

This is the process of copying a database from one server to another so that all users share the same sort of data.

11- Backup and Recovery:

Backup is the process of making an extra copy of data that you can use if the original data is lost or damaged. Recovery is the process of restoring data that has been lost, stolen or damaged in a way that makes it impossible for you to use it.

12- Concurrency Control:

This is a database management systems (DBMS) concept that is used to address conflicts with the simultaneous accessing or altering of data that can occur with a multi-user system.

13- Security and Authorization:

Database security concerns the use of a broad range of information security controls to protect databases.

By: Waleed Mousa

14- Data Consistency:

The term data consistency is a state in a database where data is consistent across all the rows of a table and across all the tables of a database.

15- Deadlock Management:

Deadlocks are a condition where two transactions cannot proceed because each holds a lock that the other needs.

16- Database Auditing:

Auditing is a feature that provides an additional layer of security by tracking the sequence of operations executed in the database.

17- Data Warehousing:

A data warehouse is a large store of data collected from a wide range of sources used to guide business decisions. This allows analysts, managers, and executives to access large amounts of information for strategic decision-making purposes.

18- ETL:

Extract, Transform, Load (ETL) is a process that involves extracting data from outside sources, transforming it to fit business needs, then loading it into the end target (database, more specifically, operational data store, data mart, or data warehouse).

By: Waleed Mousa