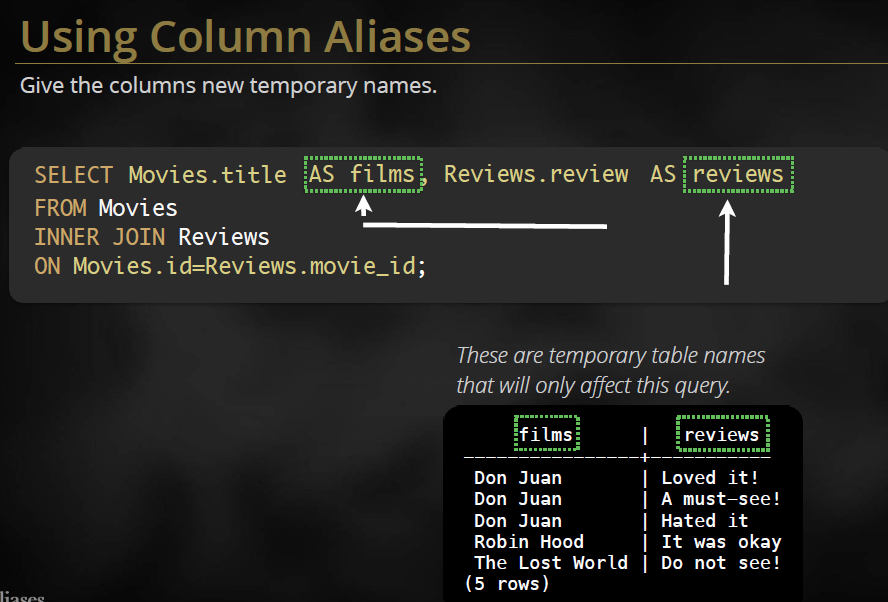
Execution Order of the SQL query

* FROM
* ON
* JOIN
* WHERE
* GROUP BY
* WITH CUBE or WITH ROLLUP
* HAVING
* SELECT
* DISTINCT
* ORDER BY
* TOP

Alias

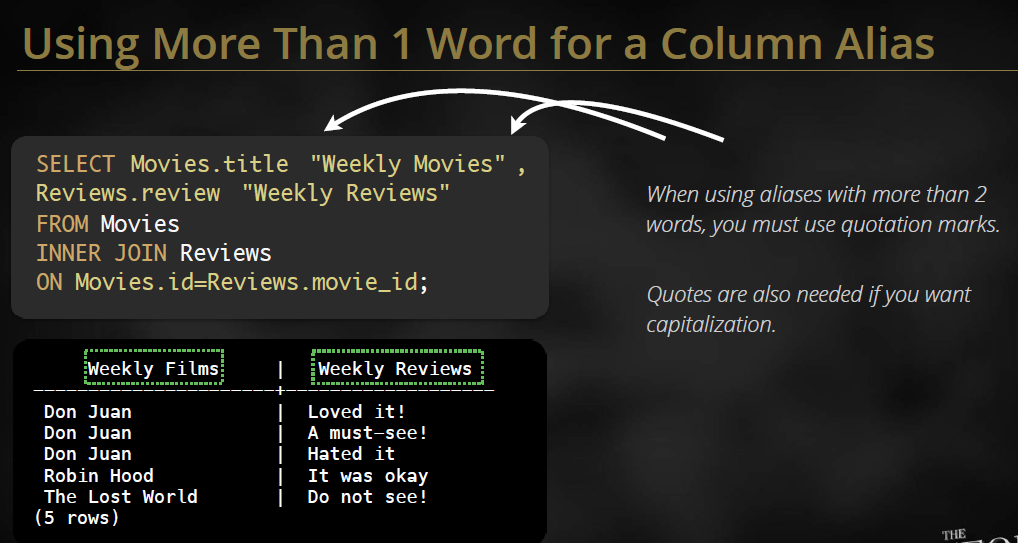
* Gives the alternative name to the column or table



* Drop the as from before the alias name



* Using more than one words for column alias names



SELECT Statement:

* The SELECT statement retrieves data from a database and returns it in the form of query results.
* The result of a SQL query is always a table of data.
* Row Selection (WHERE clause)
* It is a row wise operation
* The WHERE clause is used to specify a search condition that limits the number of rows retrieved.
* The column-names are not case-sensitive; the values of the column(s) are case sensitive.

BETWEEN, IN, LIKE

* The BETWEEN operator includes both the end values specified.
* The IN operator is used to check if a value belongs to a set of values.
* Note that BETWEEN and IN can be fully substituted with a combination of AND, OR, NOT.
* The LIKE operator is used to check for similarity of strings.
* When used with LIKE the use of “\_” refers to exactly one unknown character; “%” refers to an unknown number of unknown characters.
* List all Account s where the Bank\_Branch column has ‘o’ as the second character.

SELECT Cust\_ID, Cust\_Last\_Name, Account\_No

FROM Customer\_Details

WHERE Bank\_Branch LIKE ‘\_o%’;

IS NULL, IS NOT NULL

* The NULL value is used to indicate the absence of a value.
* A NULL value is not equal to another NULL value. The result of comparing two NULL values is NULL. It is neither TRUE nor FALSE.

ORDER BY

* ORDER BY clause can be followed by the column name or the position of the column as it appears in the SELECT statement.

Aggregate Functions / Column Functions

* SQL allows summarizing data from the database through a set of column functions. A SQL column function takes an entire column of data as its arguments and produces a single data item that summarizes the column.
* Following are some of the widely used column functions:

- SUM() : computes the total of a column

- AVG() : computes the average value in a column

- MIN() : finds the smallest value in a column

- MAX() : finds the largest value in a column

- COUNT() : counts the number of non-NULL values in a column

- COUNT (\*): counts rows of query results and does not depend on the presence or absence of NULL values in a column. If there are no rows, it returns a value of zero.

* NOTE: Rows that have a NULL value in the relevant column are ignored by all the above aggregate function except count (\*).
* Note: COUNT (Column-Name) counts the number of non-NULL values in a column whereas COUNT (\*) counts rows of query results and does not depend on the presence or absence of NULL values in a column.

GROUP BY

* The GROUP BY clause is used in a SELECT statement to collect data across multiple records and group the results by one or more columns.
* Sometimes it is required to get information not about each row, but about each group.
* If the GROUP BY clause has been used in a SELECT statement, only the grouping columns (columns on which grouping has been done) or aggregate functions can appear in the column list specified in the SELECT statement. Example:

SELECT Department, Manager\_ID, COUNT(Employee\_ID)

FROM Employee\_Manager

GROUP BY Manager\_ID, Department;

Having

* The HAVING clause is used along with the GROUP BY clause. The format of the HAVING clause is similar to the WHERE clause, consisting of the keyword HAVING followed by a search condition.
* The HAVING clause thus specifies a search condition for groups

Example:

SELECT Cust\_ID, SUM(Amount\_in\_Dollars) FROM Customer\_Loan GROUP BY Cust\_ID HAVING SUM(Amount\_in\_Dollars) > 4000.00;

* The WHERE clause can be used to select and reject the individual rows that participate in a query. The HAVING clause can be used to select and reject row groups.

Retrieval using UNION

* The UNION operation combines the rows from two sets of query results. By default, the UNION operation eliminates duplicate rows as part of its processing. Example:

SELECT Cust\_ID

FROM Customer\_Fixed\_Deposit

UNION

SELECT Cust\_ID

FROM Customer\_Loan;

* To retain duplicate rows in a UNION operation, specify the ALL keyword immediately following the word UNION. Example:

SELECT Cust\_ID

FROM Customer\_Fixed\_Deposit

UNION ALL

SELECT Cust\_ID

FROM Customer\_Loan;

There are some restrictions on the table that can be combined by a UNION operation:

* The SELECT statements combined using UNION or UNION ALL must contain the same number of columns.
* The data type of each column in the first table must be the same as the data type of the corresponding column in the second table. The data width and column name can differ.
* Neither of the two tables can be sorted with the ORDER BY clause. However, the combined query results can be sorted
* NOTE: Eliminating duplicate rows from query results is a time-consuming process, especially if the query results contain a large number of rows. If one is sure that the UNION operation cannot produce duplicate rows, one should specifically use the UNION ALL operation because the query will execute much more quickly.

Retrieval using INTERSECT

* The INTERSECT operation selects the common row from two sets of query results.

Example:

SELECT Cust\_ID

FROM Customer\_Fixed\_Deposit

INTERSECT

SELECT Cust\_ID

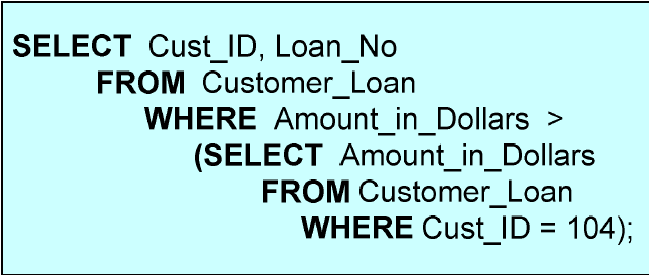
FROM Customer\_Loan;

Sub-Queries

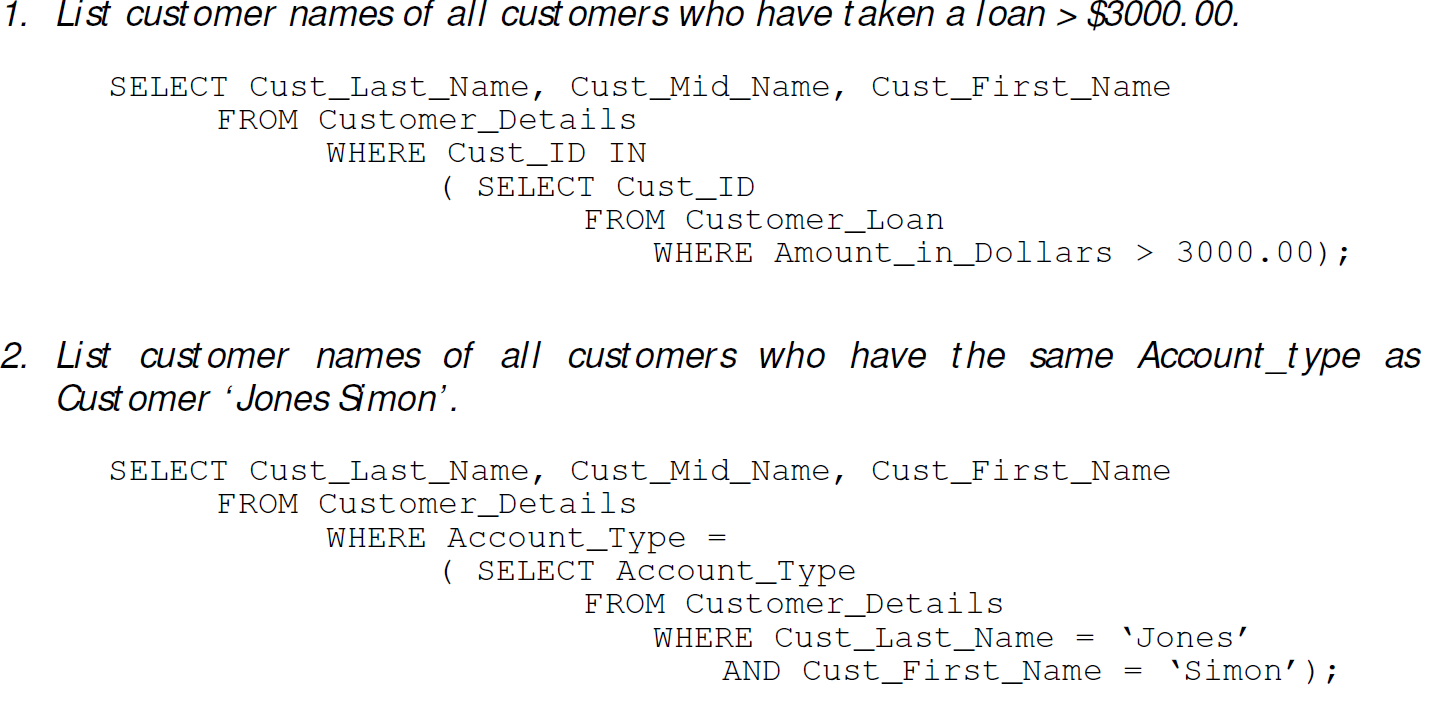
* A sub-query is a query within a query.
* The results of the sub-query are used by the DBMS to determine the results of the higher-level query that contains the sub-query.
* Usually, the sub-query appears within the WHERE or HAVING clause of another SQL statement.
* The sub-query is enclosed in parentheses, but otherwise it has a form similar to that of a SELECT statement, with a FROM clause and optional WHERE, GROUP BY, and HAVING clauses.

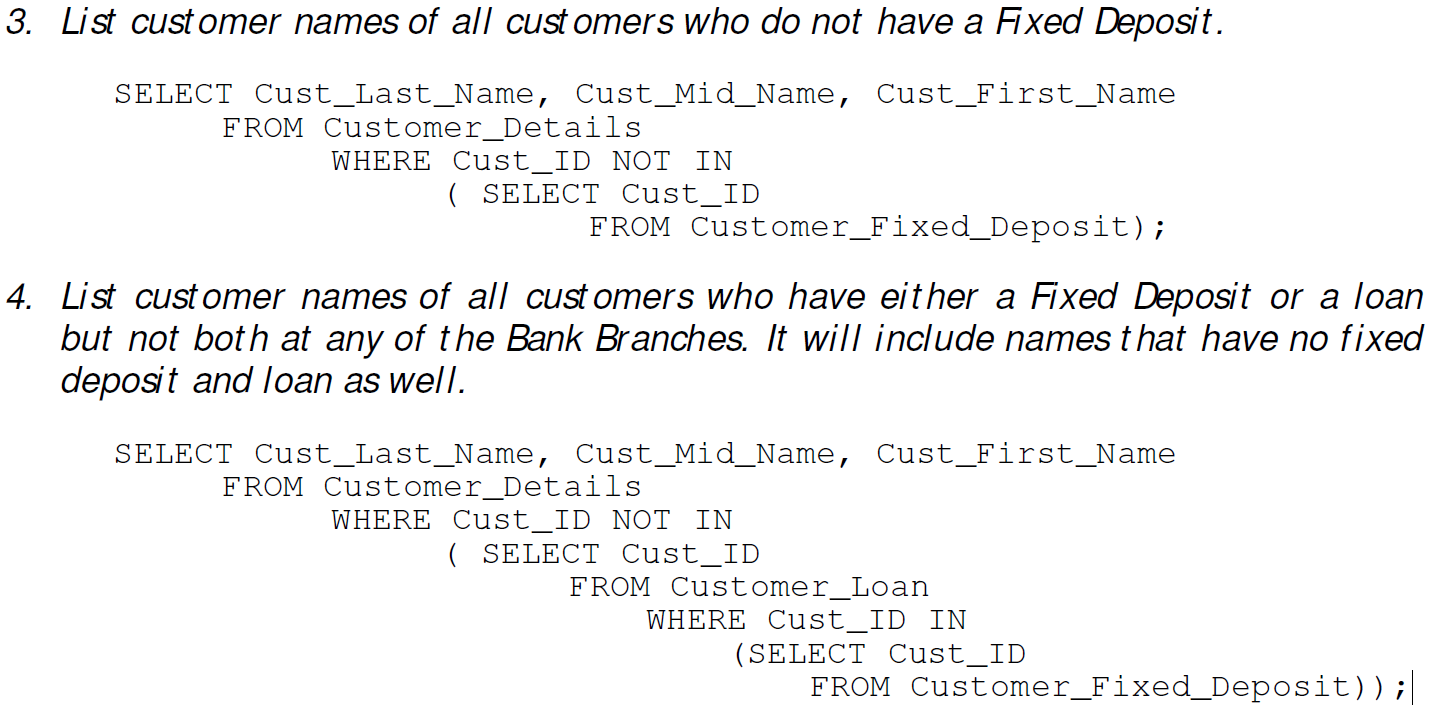
Independent Sub-Queries

* Inner Query is independent of Outer Query
* Inner Query is executed first and the results are stored
* Outer Query then runs on the stored results



Example of Sub-Query:





**JOINS**

* Join operations take two tables and return another table as a result.
* Cartesian product: A mathematical term that, when applied to relational databases, refers to the result obtained by joining all the rows of one table with all the rows of another table in every possible combination.
* SELF JOIN: Joining a table with itself is a self-join.
* INNER JOIN: An inner join between two (or more) tables is the Cartesian product that satisfies the join condition in the WHERE clause.
* Inner joins use a comparison operator like = or <> to match rows from two tables based on the values in common columns from each table.
* OUTER JOINS: An inner join provides only those values that satisfy the WHERE condition.
* An outer join is then used to retrieve the rows with an unmatched value in the relevant column.

**The Order of Execution of a SELECT statement**

* If a SELECT Statement contains a WHERE, GROUP BY, HAVING and ORDER BY CLAUSE, the order of execution is as follows:

1. The WHERE clause is applied first, and the rows for which the search condition in the WHERE clause returns a TRUE are retained.
2. Next a GROUP BY clause is applied. It will group the rows selected by the WHERE clause such that all the rows in each group have the same value for the column in the GROUP BY clause.