Chapter 1

SQL for Data Preparation

1.1 组合数据

1.1.1 连接的类型

You will learn about three fundamental joins, which are illustrated in Figure 1.1—inner joins, outer joins, and cross joins:

外连接

完全外连接将返回左表和右表中的所有行，无论连接谓词是否匹配。对于满足连接谓词的行，两行将被组合起来，就像内部连接一样。对于不满足的行，两个表中的每一行都将被选择为单独的行，并为另一个表中的列填充 **NULL**。完全外连接是通过使用 **FULL OUTER JOIN** 子句并后跟连接谓词来调用的。

1.1.2 Subqueries

If a query only has one column, you can use a subquery with the **IN** keyword in a **WHERE** clause.

1.1.3 Unions

Please note that there are certain conditions that need to be kept in mind when using **UNION**. Firstly, **UNION** requires the subqueries to have the same number of columns and the same data types for the columns. If they do not, the query will fail to run. Secondly, **UNION** technically may not return all the rows from its subqueries. **UNION**, by default, removes all duplicate rows in the output. If you want to retain the duplicate rows, it is preferable to use the **UNION ALL** keyword.

1.1.4 Common Table Expressions(CTEs)

CTEs are simply a different version of subqueries. CTEs establish temporary tables by using the **WITH** clause. The one advantage of CTEs is that they can be designed to be recursive. **Recursive**

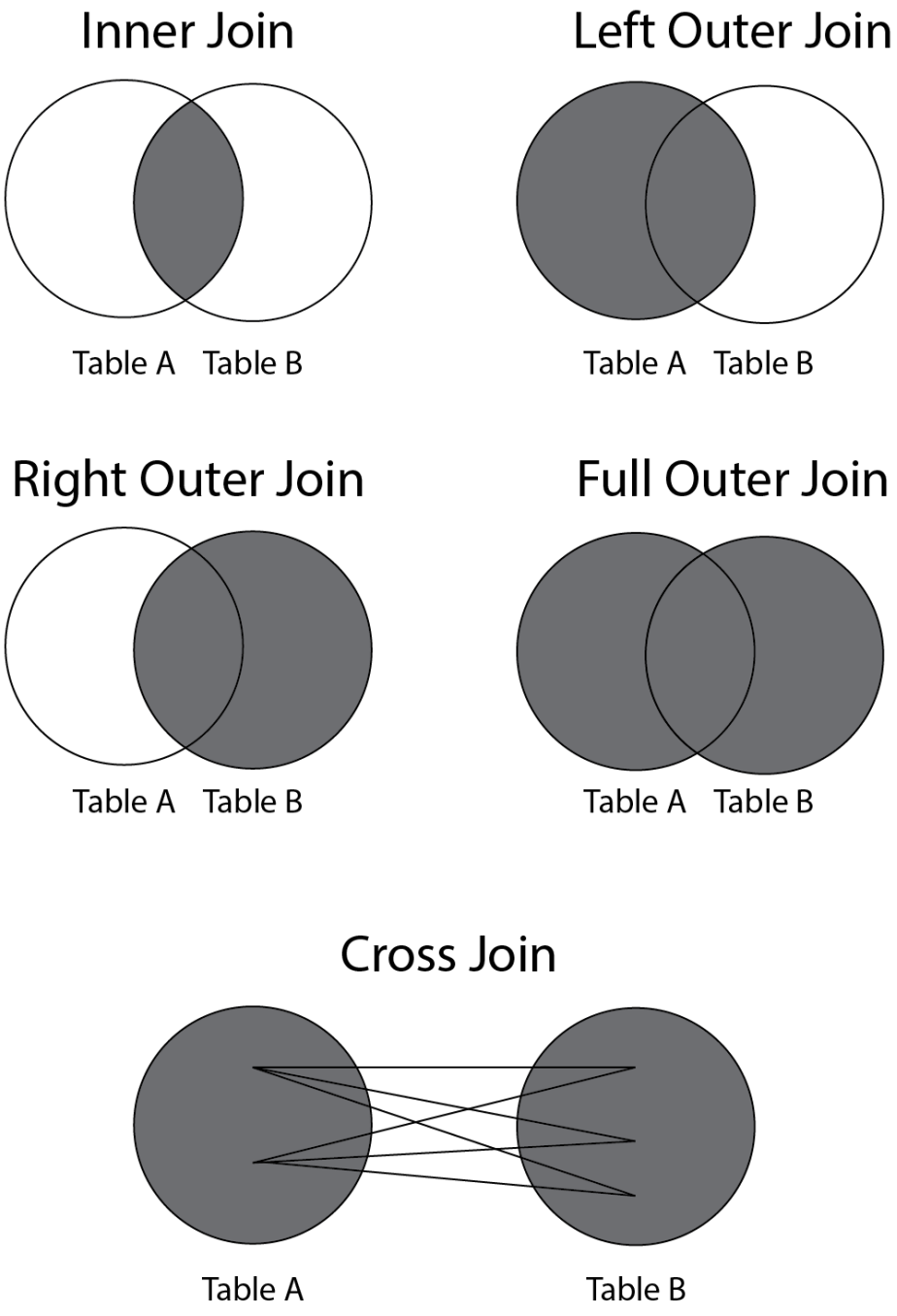

图 1.1: Major types of joins

CTEs can reference themselves. Because of this feature, you can use them to solve problems that other queries cannot.

1.2 Cleaning Data

1.2.1 The CASE WHEN Function

CASE WHEN is a function that allows a query to map various values in a column to other values.

1.2.2 The COALESCE Function

Another common requirement is to replace the NULL values with a standard value. This can be accomplished easily by means of the COALESCE function. COALESCE allows you to list any number of columns and scalar values, and, if the first value in the list is NULL, it will try to fill it in with the second value. The COALESCE function will keep continuing down the list of values until it hits a non-NULL value. If all values in the COALESCE function are NULL, then the function returns NULL.

1.2.3 The NULLIF Function

NULLIF is used as the opposite of COALESCE. While COALESCE is used to convert NULL into a standard value, NULLIF is a two-value function and will return NULL if the first value equals the second value.

1.2.4 The LEAST/GREATEST Functions

Two functions that come in handy for data preparation are the LEAST and GREATEST functions. Each function takes any number of values and returns the least or the greatest of the values, respectively. The simple use of this variable would be to replace the value if it is too high or low.

1.2.5 The Casting Function

To change the data type of a column, you simply need to use the **column::datatype** format, where column is the column name and datatype is the data type you want to change the column to.

Please note that not every data type can be cast to a specific data type.

1.3 Transforming Data

1.3.1 The DISTINCT and DISTINCT ON Functions

When looking through a dataset, you may be interested in determining the unique values in a column or group of columns. This is the primary use case of the DISTINCT keyword.

Another keyword related to DISTINCT is DISTINCT ON. Now, DISTINCT ON allows you to ensure that only one row is returned, and one or more columns are always unique in the set.

Chapter 2

Aggregate Functions for Data Analysis

2.1 Aggregate Functions

2.2 Aggregate Functions with the GROUP BY Clause

2.2.1 Ordered Set Aggregates

Function	Explanation
count(columnX)	Counts the number of rows in columnX that have a non-NULL value
count(*)	Counts the number of rows in the output table
min(columnX)	Returns the minimum value in columnX. For text columns, it returns the value that would appear first alphabetically
max(columnX)	Returns the maximum value in columnX
sum(columnX)	Returns the sum of all values in columnX
avg(columnX)	Returns the average of all values in columnX
stddev(columnX)	Returns the samples standard deviation of all values in columnX
var(columnX)	Returns the samples variance of all values in columnX
regr_slope(columnX, columnY)	Returns the slope of linear regression for columnX as the response variable and columnY as the predictor variable
regr_intercept(columnX, columnY)	Returns the intercept of linear regression for columnX as the response variable and columnY as the predictor variable
corr(columnX, columnY)	Calculates the Pearson correlation between columnX and columnY in the data

Function	Explanation
mode()	Returns the value that appears most often. In the case of a tie, it returns the first value in order
percentile_cont()	Returns a value corresponding to the specified fraction in the ordering, interpolating between adja
percentile_disc()	Returns the first input value whose position in the ordering equals or exceeds the specified fractio

Chapter 3

3.1 Window Functions

3.1.1 The Basics of Window Functions

Figure 3.1 the dataset is ordered using customer_id, which happens to be the primary key. As such each row has a unique value and forms a value group. The first value group, without any row before it, forms its own window, which contains only the first row. The second value group's window will contain both itself and the row before it, which means the first and second row. Then the third value group's window will contain itself and the two rows before it, and so on and so forth. Every value group has its window. Once the windows are established, for every value group, the window function is calculated based on the window. In this example, this means COUNT is applied to every window. Thus, value group 1 (the first row) gets 1 as the result since its Window 1 contains one row, value group 2 (the second row) gets 2 since its Window 2 contains two rows, and so on and so forth. The results are applied to every row in this value group if the group contains multiple rows. Note that the window is used for calculation only. The results are assigned to rows in the value group, not assigned to the rows in the window.

图 3.1: Windows for customers using COUNT(*) ordered by the customer_id window query

表 3.1: Statistical window functions

Name	Description
row_number	Number the current row within its partition starting from 1
dense_rank	Rank the current row within its partition without gaps
rank	Rank the current row within its partition with gaps
lag	Return a vale evaluated at the row that is at a specified physical offset row before the current row within the partition
lead	Return a value evaluated at the row that is offset rows after the current row within the partition
ntile	Divide rows in partition as equally as possible and assign each row an integer starting from 1 to the argument value

3.2 Statistics with Window Functions

Note

One question regarding RANK() is the handling of tied values. RANK() is defined as the rank of rows, not the rank of values. For example, if the first two rows have a tie, the third row will get 3 from the RANK() function. DENSE_RANK() could also be used just as easily as RANK(), but it is defined as the rank of values, not the rank of rows. In the example above, the value of DENSE_RANK() for the third row will be 2 instead of 3, as the third row contains the 2nd value in the list of values.