# Data Modeling with Snowflake: A Practical Guide to Accelerating Snowflake Development Using Universal Data Modeling Techniques

## Part 3: Solving Real-World Problems with Transformational Modeling

### Chapter 18: Appendix and Extras

* This book follows a spiral model, starting from a high-level overview of modeling + rising in winding loops, revisiting its core elements in ever-greater depth + complexity
* **The difficulty of the technical *how* is alleviated by the understanding of the conceptual *why***
* Some exercises had to be excluded b/c they did not cleanly fit within the framework of the chapters

#### Technical Requirements for Local Snowflake Work

* The scripts used to instantiate + load the examples in this chapter are available in the following GitHub repo: <https://github.com/PacktPublishing/Data-Modeling-with-Snowflake/tree/main/extras>

#### The Exceptional Time Traveler

* “What changed?” is a question that’s been asked since the first DML operations were performed on a database
* While **Snowflake’s built-in table change tracking + Streams** can help answer this question, they are **not enabled on tables by default**
* However, even in Snowflake’s Standard edition, **all tables have a default time travel data retention period of one day**
* The **time travel feature can be combined w/ the EXCEPT set operator to isolate + compare any changes made**
* The following exercise uses a randomly generated filter when selecting which records to update to make things interesting
* The only way to solve the mystery is to use Snowflake’s exceptional time-traveling powers
* The exercise is explained in the file titled *01\_exceptional\_time\_travel.sql*
* <https://docs.snowflake.com/en/user-guide/data-time-travel>
* <https://quickstarts.snowflake.com/guide/getting_started_with_time_travel/index.html#0>
* <https://docs.snowflake.com/en/sql-reference/operators-query>

#### The Secret Column Type Snowflake Refuses to Document

* Snowflake’s co-founders + chief architects, Benoit Dageville + Thierry Cruanes, spent many years working at Oracle
* In fact, Oracle’s influence can be seen in many of the SQL constructs + functions that Snowflake supports
* One such example is the concept of **virtual columns**, which **straddle the line between physical and transformational modeling** **(between table and view)**
* Virtual columns ***look like* normal table columns, but their values are *derived* rather than stored on disc**
* They are **an efficient way to embed simple business rules + transformational logic in a table w/out the overhead of maintaining views + incurring storage costs**
* Virtual columns **can be defined through constants or transformational expressions such as the DEFAULT** **column operator**
* Strangely, they are *not* mentioned in the CREATE TABLE documentation at the time of writing
* <https://docs.snowflake.com/en/sql-reference/sql/create-table>
* However, **references to Snowflake users taking advantage of them can be found in the community forums as far back as 2018**
* <https://community.snowflake.com/s/question/0D50Z00008ixGQKSA2/does-snowflake-supports-computed-columns-in-a-table-while-creating-table>
* For an overview of the advantages of using virtual/computed columns in tables + how they differ from column defaults, see the example in the file titled 02\_secret\_virtual\_columns.sql

#### Read the Functional Manual

* Snowflake’s technical documentation is among the clearest and most informative (virtual columns aside)
* Paying attention to the **usage notes** + **best practices** or simply scrolling down the list of available functions can help elevate your SQL game + discover new features when first starting, + it can continue to pay dividends
* For example, note that for ORDER BY (<https://docs.snowflake.com/en/sql-reference/constructs/order-by>), there are the NULLS FIRST and NULLS LAST keywords which override the default ordering of NULL when arranging in ASC or DESC
* **Check the documentation periodically to help spot new features and functionality**