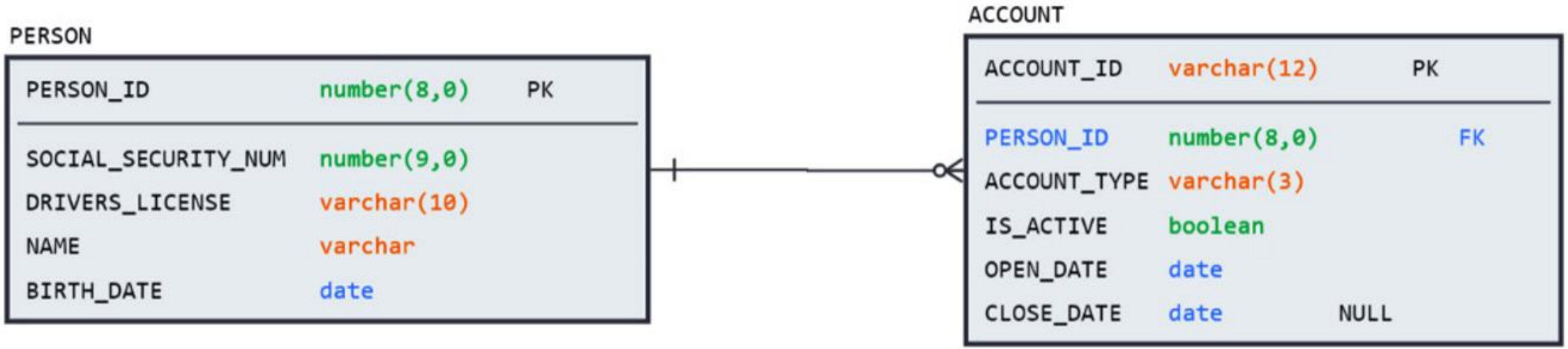
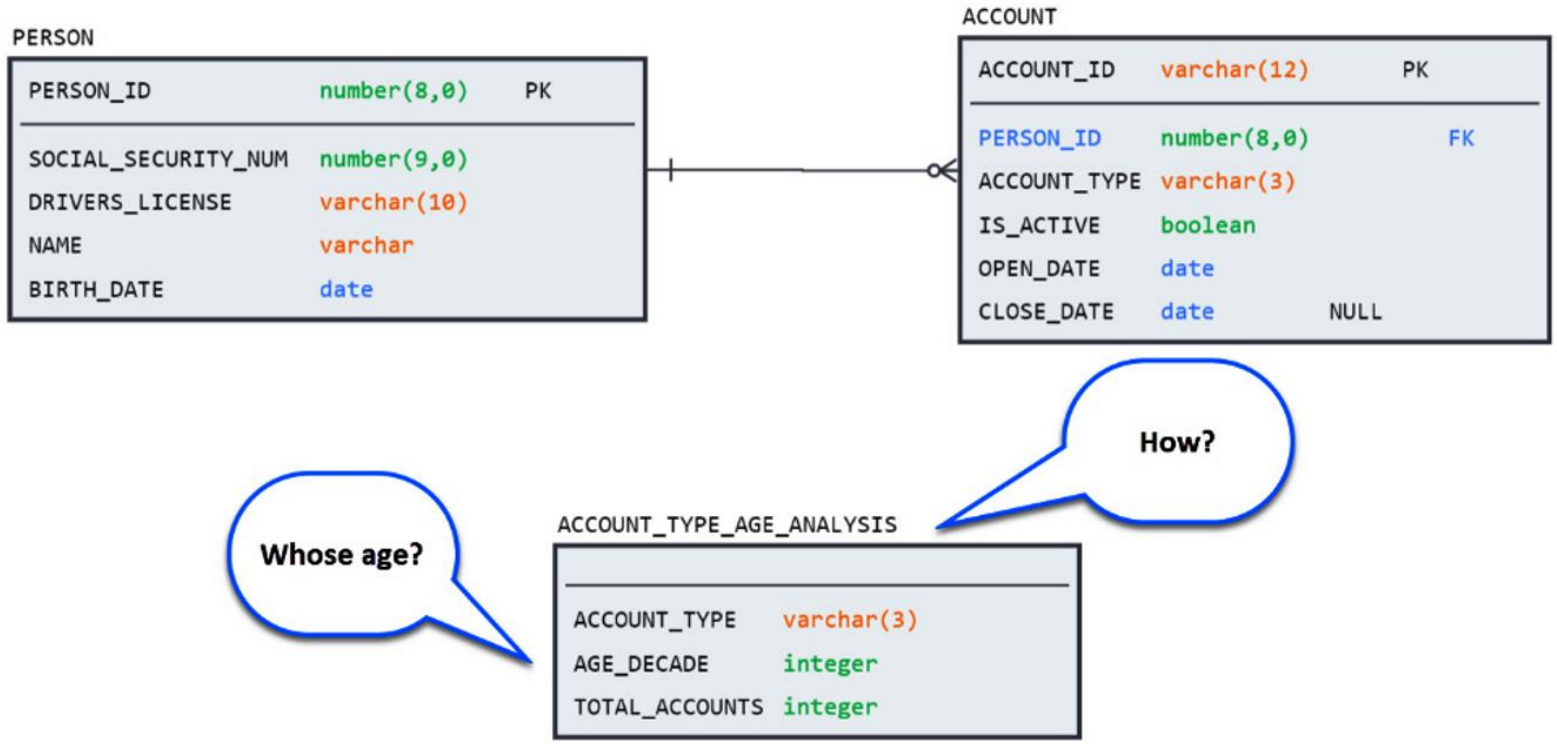
1. What modelling looks like in operational systems? Let's understand it by an example.



The physical diagram in above serves both as a blueprint for declaring the required tables and a guide to understanding their business context. Without diving deep we can infer a lot of information from above diagram. For example, A person is uniquely identified by an eight-digit identifier (the primary key) and must have a Social Security number (SSN), driver's license, name, and birth date.
The one-to-many relationship between the two tables establishes that while a person does not necessarily need to have an account created, an account must belong to just one person. These details, combined with the list of attributes, data types, and constraints, not only dictate what kinds of data can be written to these tables but also provide an idea of how the business operates. So, how does this differ in analytical databases?

2. What modelling looks like in analytical systems ?

In a data warehouse scenario, the **PERSON** and **ACCOUNT** tables would not be defined from scratch they would be extracted from the source in which they exist and loaded bringing both structure and data into the process. Then, the analytical transformations begin in answer to the organization's business questions. This is a process known as **Extract Transform Load** (**ETL**).



Suppose the management team wanted to analyze which age groups (by decade) were opening which account types and they wanted to store the result in a separate table for an independent analysis.

The following diagram shows the resulting relational model of an object obtained through transformational analysis but provides no business context

Although physical modelling could describe such a table (in pervious image) containing the account type with age and count of accounts as integers—such a model would fail to communicate the most relevant details, presented here:

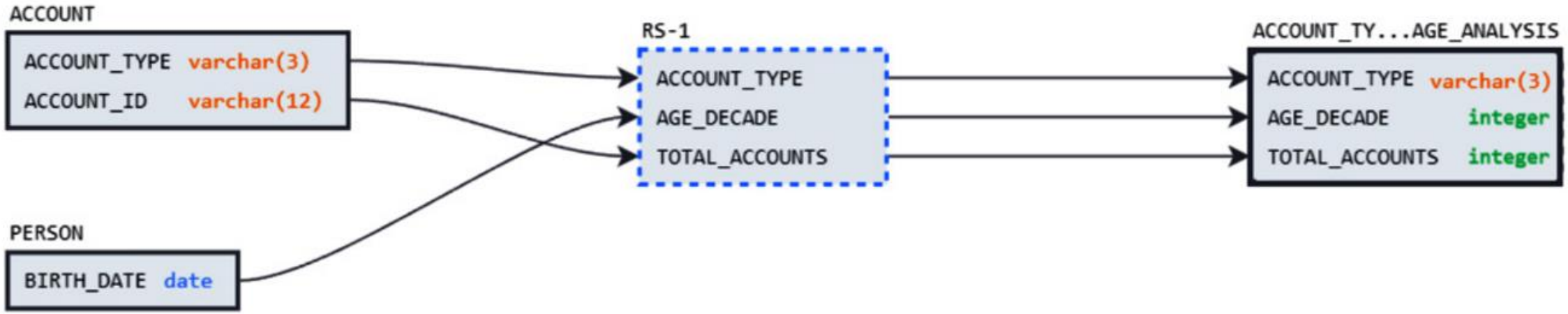
- The logic used to perform the analysis
- The relationship between the source tables and the output

The business requirement for ACCOUNT_TYPE_AGE_ANALYSIS in this example purposely excludes the source key fields from the target table, preventing the possibility of establishing any relational links. However, the relational model still serves a vital role: it tells us how the sources are related and how to join them correctly to produce the required analysis.

The logic could then be constructed by joining **PERSON** and **ACCOUNT**, as shown here:

```
CREATE TABLE account_types_age_analysis AS
SELECT  a.account_type,   ROUND(DATEDIFF(years, p.birth_date, CURRENT_DATE()), -1 ) AS age_decade,
  COUNT(a.account_id) AS total_accounts   FROM account AS  a
  INNER JOIN person AS p ON a.person_id = p.person_id
GROUP BY     1, 2;
```

Although there is no relational connection between **ACCOUNT_TYPE_AGE_ANALYSIS** and its sources, there is still a clear dependency on them and their columns. Instead of using ERDs, which convey entities and relationships, transformational pipelines are visualized through a lineage diagram.



Follow : Avinash Sharma | LinkedIn for more Data Engineering Related Posts.

Get Complete Interview Resource For Snowflake, SQL, python and AWS: (Including Hands on Lab Snowflake and Python
Course, Practical questions for snowflake):
https://topmate.io/avinash_sharma/869015?coupon_code=GETOFF25

Get Snowflake interview resources:
https://topmate.io/avinash_sharma/501313

Pivot to Data Engineering. Get 1:1 Mentorship:
1:1 End-End Mentorship Snowflake Data Engineering. with Avinash Sharma (topmate.io)