**Database Schema:**

It represents the logical storage of the data.

Types of schema:

1. Physical schema
2. Logical schema
3. View schema

**1. Physical Schema**

1. Describes the low-level storage details of the database, such as file paths, storage structures, and indexing.
2. Specifies how data is stored on the hardware, including data blocks, file organization, and access paths.
3. Deals with storage allocation techniques, like hashing, partitioning, and clustering.
4. Focuses on optimizing performance by organizing data in ways that minimize storage space and speed up access.
5. Managed by the database administrator, as changes often require expert knowledge of hardware and database internals.

**2. Logical Schema**

1. Defines the structure of the database on a conceptual level, independent of physical storage.
2. Includes entities, relationships, constraints, and data types, focusing on the "what" rather than the "how" of data.
3. Used by developers to understand and design the database model, aligning it with business requirements.
4. Supports database independence, allowing changes to the physical schema without affecting the logical structure.
5. Is crucial for data integrity and consistency, as it sets the rules that data must follow within the database.

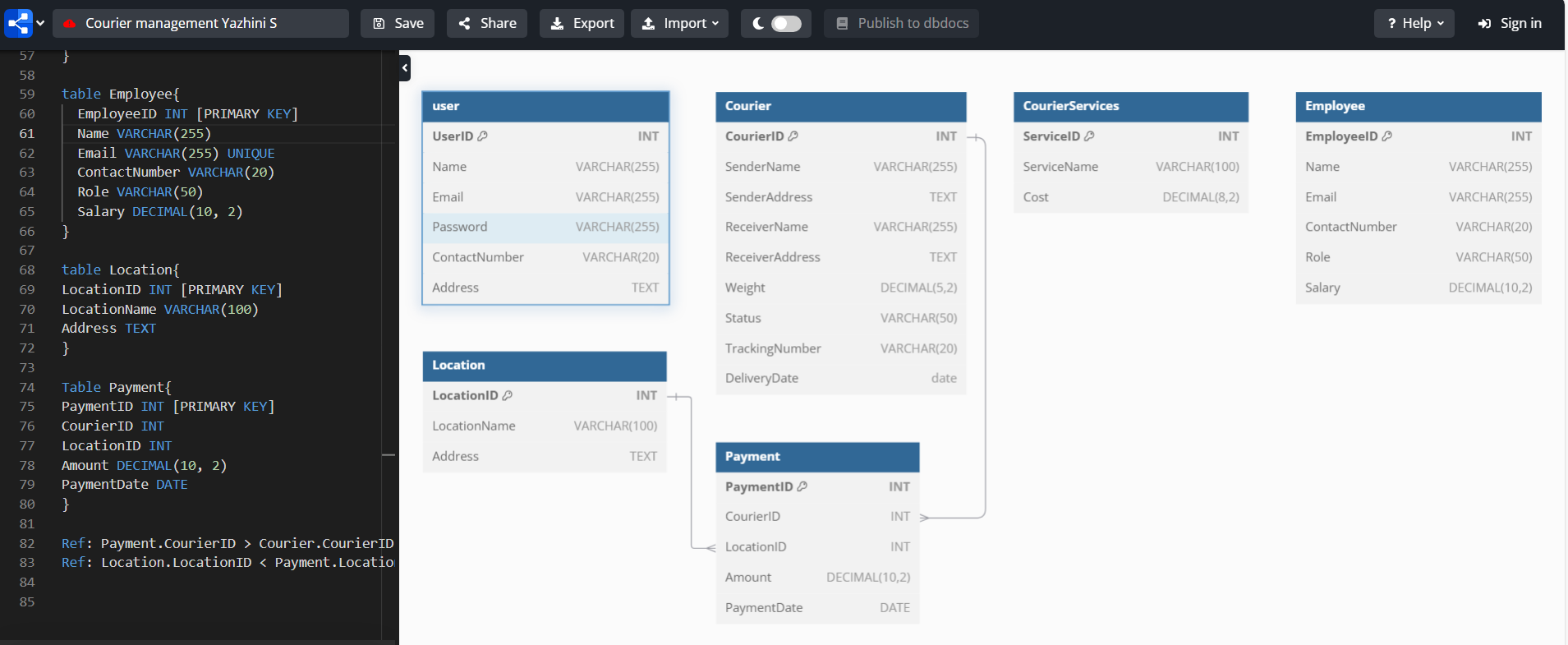
**3. View Schema**

1. Defines customized views or virtual tables for users, derived from the logical schema.
2. Provides specific perspectives of the data, often restricting access to sensitive information for security.
3. Allows users to query complex data in a simplified format, improving usability for non-technical users.
4. Supports data abstraction by hiding details, presenting only relevant data fields and rows as per user requirements.
5. Enables multiple views for different user roles, ensuring data privacy while allowing shared access across departments or teams.

**Data base:**

It is an organized collection of data stored and managed electronically, allowing for easy access, management, and updating of information. Databases are used to store and retrieve data in a structured manner, enabling efficient querying, reporting, and analysis.

**ER Diagram: example tried**

****

**Database instance:**

Gives us a backup option.

**Basic Structure:**

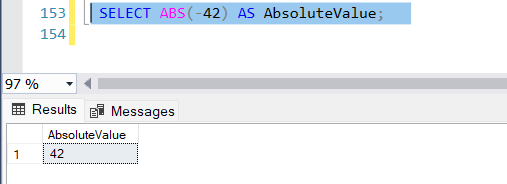
select \* from employee;

**Some SQL Functions:**

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Example** | **Description** |
| abs | SELECT abs (-77) | Returns 77, an absolute value. |
| cos,sin,tan,cot | SELECT sin(1.5) | Returns the angle in radians. |
| ceiling | SELECT ceiling (14.45) | Returns 15, the smallest or greater than or equal to the specified value. |
| exp | SELECT exp (4.5) | Returns 90.017, the exponential value of the specified parameter. |
| floor | SELECT floor (15.55) | Returns 15, the largest or less than or equal to the specified value. |
| log | SELECT log (5.4) | Returns 1.6869, the logarithm of the parameter. |

**Examples done:**

**1.**

****

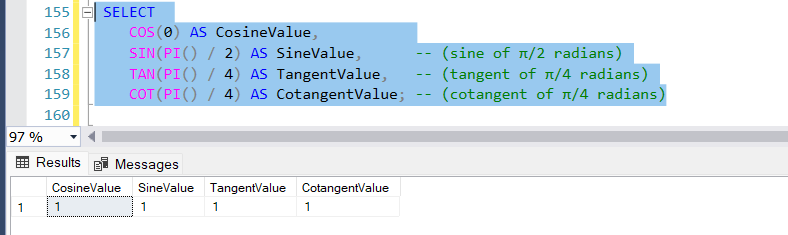
**2.** SELECT

COS(0) AS CosineValue, -- (cosine of 0 radians)

SIN(PI() / 2) AS SineValue, -- (sine of π/2 radians)

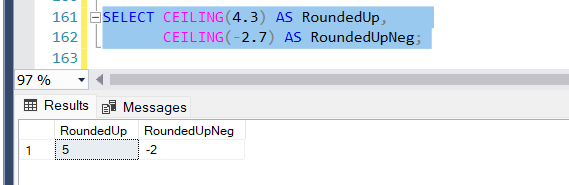
TAN(PI() / 4) AS TangentValue, -- (tangent of π/4 radians)

COT(PI() / 4) AS CotangentValue; -- (cotangent of π/4 radians)

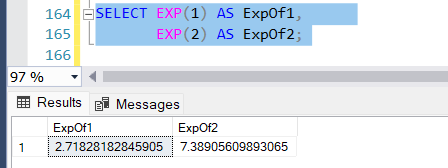


**3.**SELECT CEILING(4.3) AS RoundedUp,

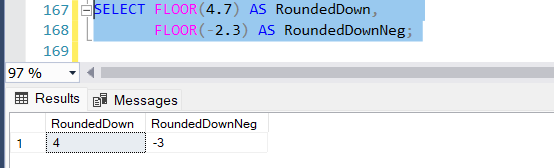
CEILING(-2.7) AS RoundedUpNeg;



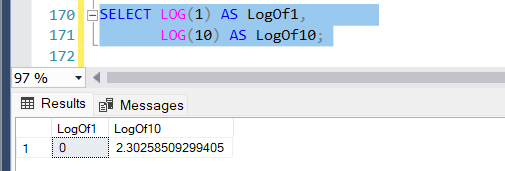
**4.**Exponential:



**5.**Floor:



**6.**Log:



**7.**ROUND:

This function helps us to round a value to a particular decimal.

Syntax or Querry:

SELECT EmployeeID, ROUND(Salary, 2) AS [Hourly Rate]

FROM employee;

