# Module 3: Insert Function. ChatGPT write back to Azure SQL Database

The module is dedicated to presentation of different ways how the initial logic of communication with your SQL DB data might be extended. For the purpose of this Hackathon, we have worked with INSERT function (flow is presented below). It is worth mentioning that the function was implemented as a trial by Hackathon presenters. Important for us was to show to the user that any ideas might be realized and built upon already provided solutions.

## Use Case Scenario: Recording Scrap Data in AdventureWorks2019 Database

**Background:** AdventureWorks is a manufacturing plant that produces a variety of products. To maintain quality control and minimize waste, the plant has a Scrap table in its AdventureWorks2019 database to record and analyze scrapped parts and materials. (Note: there are special systems that collect information about the scrapped parts on the production lines at plants. The presented Use Case is just an example of insert functionality).

**Scenario:** In the AdventureWorks manufacturing plant, the Scrap Redemption Center plays a crucial role in documenting and managing scrap data. Here's how the process works:

1. **Collection of Scrap**: Throughout the manufacturing process, workers collect any defective or rejected parts, sub-assemblies, or materials. These items are placed in designated bins or containers for later processing.
2. **Identification and Sorting**: Workers in the Scrap Redemption Center carefully inspect each piece of scrap to determine its type and the reason for rejection. This step helps in categorizing and prioritizing the scrap data.
3. **Data Entry**:

**a. Type and Description**: The worker enters information about the type of scrap (e.g., metal, plastic, electronics) and provides a brief description of the item (e.g., "Gouge in metal," " Drill size too large").

For checking the possible Scrap reasons, you can query ScrapReason table.

A screenshot of a computer

Description automatically generatedThe answer will be

Here is the query running behind your request

A screen shot of a computer

Description automatically generated

b. **Quantity**: The quantity of each type of scrap is recorded. For instance, if there are ten defective wiring harnesses, this is noted.

c. **Reason for Rejection**: A drop-down menu or predefined list of reasons for rejection is available for selection. Common reasons include "dimensional error," "material defect," or "assembly issue."

d. **Date and Time**: The date and time of each entry are automatically recorded by the system to maintain a chronological record.

1. **Database Input**: The worker uses a computer terminal connected to the AdventureWorks2019 database. Using a simple user interface, they input the gathered data into the staging table from where it will be redirected to the appropriate table in AdventureWorks2019 database. The system validates entries to ensure accuracy.
2. **Confirmation and Verification**: After entering the data, the worker reviews it for accuracy, making any necessary corrections.
3. **Submission**: Once all the scrap data for a particular shift or time frame is accurately recorded in the stagging tables, the worker submits the data into the AdventureWorks2019 database.
4. **Reporting and Analysis**: Management and quality control teams have access to the AdventureWorks2019 database. They use it to generate reports and conduct analyses to identify trends, pinpoint recurring issues, and make informed decisions about process improvements.

## Staging table

For the purposes of maintaining data integrity, it might be recommended to use a staging table that will be used for inserting and storing of the initially inserted data. It serves as an intermediate storage area for data before moving it to its final destination. Before loading the data to its required destination, it often needs to be validated, cleaned, and transformed. Staging tables might be used as a safe environment to perform insert operations by “frontline” people (for example, plant workers) without deep SQL Server knowledge. The next phase of data cleaning, correct transformation and distribution should be performed by SQL experts.

Let’s continue with our use case and create a staging table for the previous scenario. Imagine that you’re a frontline worker who needs to insert the data about the scrap parts. You will know information about the product you are working with, scraps quantity and the reason of scrap parts.

CREATE TABLE Staging\_WorkOrderScrap (

ScrapID INT IDENTITY(1,1) PRIMARY KEY, -- Auto-incremented unique identifier for each scrap record

ProductID INT, -- Foreign key to identify the product associated with the scrap

ScrappedQty INT, -- Quantity of scrapped parts

ScrapReasonID INT, -- Foreign key to specify the reason for scrap

ScrapDate DATETIME DEFAULT GETDATE() -- Date and time of scrap (default to current timestamp)

);

We can check if the table is already there:

A screenshot of a computer

Description automatically generated

Now we are fine to add our first information about the scrap parts.

A screenshot of a computer

Description automatically generated

Let’s check the result of our performed insert together:

A screenshot of a computer

Description automatically generated

The data was inserted successfully and might be further adjusted and transferred to the correct table by SQL Server Experts.

Without the usage of stagging tables, it is important to be careful with inserting the data to existing table due to the possible harm to the data integrity.

## Issues with Insert Function

A screenshot of a computer

Description automatically generatedNow let’s imagine that you’re the one who has to analyze the produced scrap work item and enter the data directly to AdventureWorks2019 database without the usage of Staging tables. You have to include the information about the scrapped item in the system (Database: [AdventureWorks2019].[Production].[WorkOrder])

Now let’s verify the input in the database with SSMS (or perform the same with our Streamlit application):

A screenshot of a computer

Description automatically generated

The work item was inserted.

A screenshot of a computer

Description automatically generated

In the process of implementation you can observe the following issues that are related to some specific rules that should be applied to different columns:

* ERROR: The column cannot be modified because it is either a computed column or is the result of UNION operator.

After this error, the next automatically performed query skipped the column related to this error.

* ERROR: The column does not allow nulls.

This error was also automatically resolved by inserting the data which was generated automatically by the system.

As you can see, the data quality without the usage of stagging tables might require deep SQL expertise in order to understand and handle the data correctly.

Besides it, let’s have a look at the point with primary and foreign keys in the tables.

As you know the primary key is used as a unique ID and ensures that each row in the table is distinct and can be identified easily. While the foreign key might be used as a reference to another table’s primary key which allows to create a connection between two tables. It allows one table to refer to data in another table.

Let’s check the inserted data.

For that, we can use WorkOrderID (primary key in this table) and/or ProductID (foreign key in this table).

We can use our SQL Query Writing Assistant to check which tables can be compared for WorkOrderID information.

A screenshot of a computer

Description automatically generated

Now let’s check the value you have inserted. For querying the data please use WorkOrderID that you have inserted:

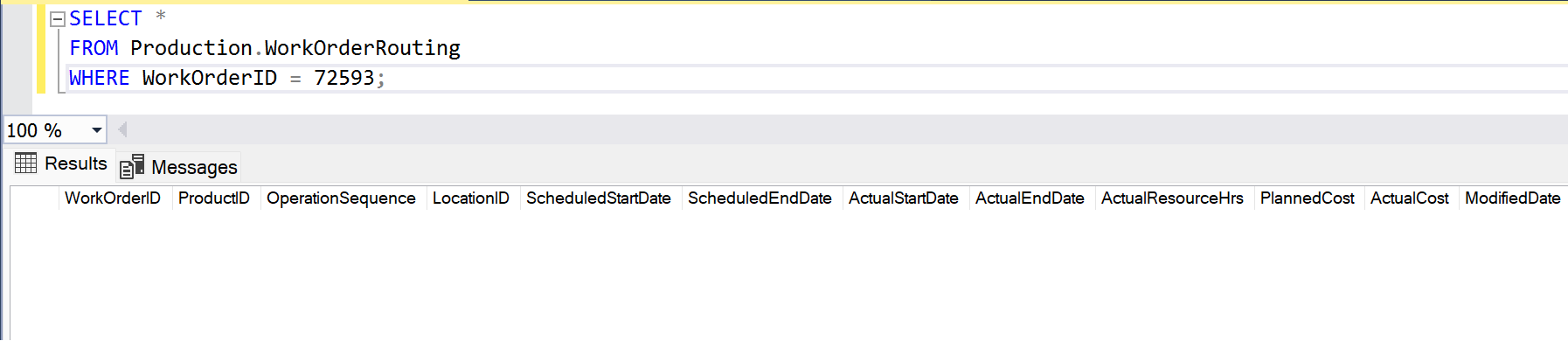
A close up of a box

Description automatically generated

Now let’s check WorkOrderRouting table where WorkOrderID

A close up of a message

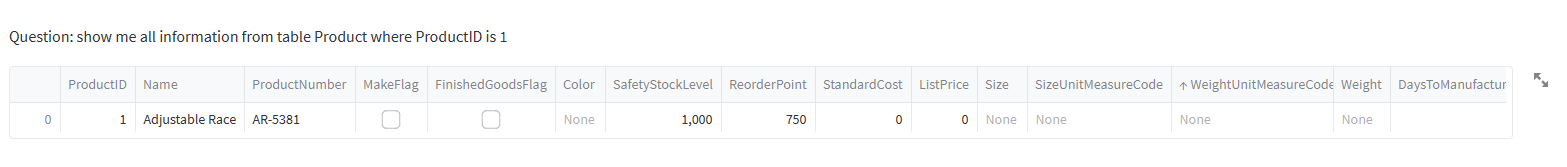
Description automatically generated

Just to verify that there is no issue, let’s check our table in SSMS to be sure that the information is correct.

As you can see, no items where found with the number “72593” as a WorkOrderID.

In our table we also use a Foreign Key “ProductID” which should serve as a UniqueID in the Product table.

For our inserted value, the automatically generated ProductID is 1. Let’s compare with the value in table Product.



As we can see, product described here does not correspond to the work item we have inserted before.

Since we have already mentioned that the usage of foreign and primary keys are important for the establishment of relationships between the tables, inconsistencies in primary and foreign keys might cause serious challenged in database management, such as data integrity problems or potential referential integrity violations.