**Business Statement:** Unconstrained Loss Model (ULM) scores leverage data science models to predict frequency, severity, and pure premium for several coverages on the policy term level. These scores would enable our Product Analytics team to more accurately predict future losses and it would give us another tool for assessing risk associated with new policies and customers.

* 1. **Recommendation for Joining Auto ULM to Diagnosable Datasets**

Given that these scores are unique on the policy term level, our team has several options for adding these fields to our diagnosable datasets. The two main datasets in consideration are the Auto Char Ref table and the Auto Operational Loss table. We want to incorporate these scores into operational loss because ULM scores would enhance our ability to predict future losses and conduct analysis on how policies are performing over time. While I can understand the benefits for adding these scores to Char Ref, I personally recommend not adding these scores to it. Our team is currently in the process of cleaning the diagnosable datasets and to my knowledge, Char Ref is used to store general information about all our diagnosable datasets. Since ULM scores are only available for a set number of policy period IDs, adding these niche fields to Char Ref would increase clutter and data redundancy. By adding these scores only to Auto Operational Loss, we will still be able to answer any business-related questions while optimizing computing resources and space in our tables.

* 1. **Logic for Joining ULM to Auto Operational Loss**

When joining the ULM view to the Auto Operational Loss table, there was some extra logic to figure out. Our auto operational loss table is broken down by policy ID, policy period ID, coverage type, and earned month. Since ULM scores are unique to just a policy period ID, we can’t simply join these two tables on that attribute because operational loss is split up into a finer grain of detail. The asset owner, Carsten Lingemann, instructed that we take the earned exposure of a record as a percentage for a given policy period and multiply it by the ULM scores. By doing this, we can break down each metric into a finer grain and when you sum up all the values for a ULM field over a policy period id, you will get the same value that is stored in the ULM view. The official GitHub repository and the contact information for the ULM asset owner is listed below:

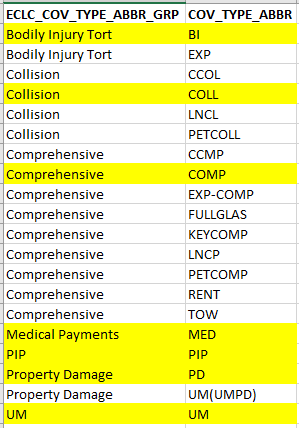
*Snowflake View:* "DSC\_PLDS\_DB"."APP\_AUTOMATA\_PRD"."PREVAIL\_AUTO\_ULM\_POL\_PRD\_AGG\_VW"

*ULM GitHub:* <https://github.thehartford.com/HIG/pl_ulm_scoremart/wiki>

*ULM Asset Owner:* [Carsten.lingemann@thehartford.com](mailto:Carsten.lingemann@thehartford.com)

* 1. **Mapping Table for Auto Operational Loss Coverages**

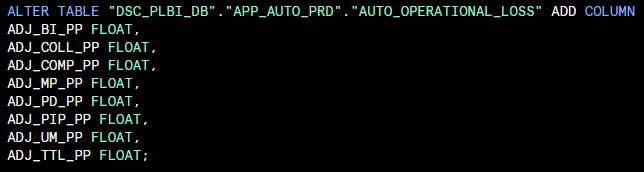
On the ULM GitHub page, it is stated that current ULM models only predict pure premium, frequency, and severity for seven different auto coverage types. These coverages are BI (Bodily Injury), COLL (Collision), COMP (Comprehensive), MP (Medical Payments), PD (Property Damage), PIP (Personal Injury Protection), and UM (Uninsured Motorist). We had to map these scores to certain records in our Auto Operational Loss table, since our table has 26 unique coverage types. Carsten created the following table to map the coverage types in operational loss to the seven coverages in ULM:



* 1. **Queries for Integrating ULM into Auto Operational Loss**

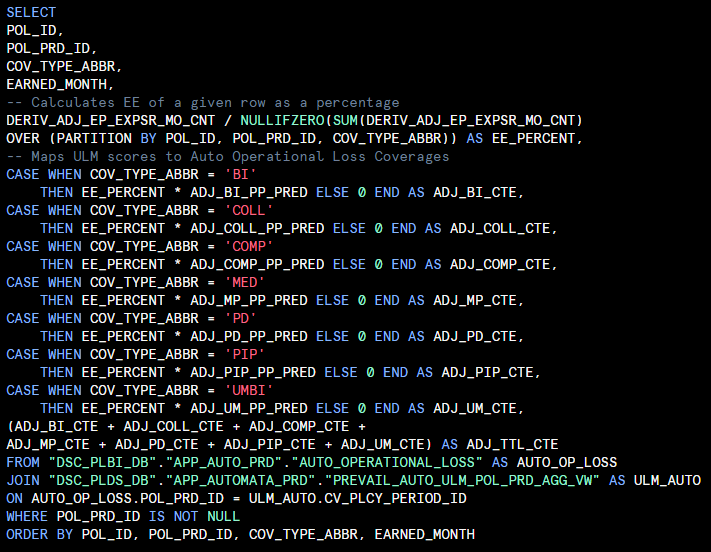
Now that the mapping and integration logic has been established, I have included the code that we can use to add ULM to our auto operational loss dataset. For any questions regarding code implementation, please contact Peter Alonzo. See SQL acronym definitions on Page 5.

DDL Query for Adding ULM Columns



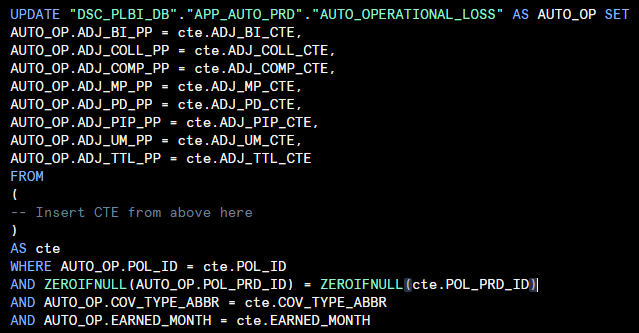
Explanation: We find it necessary to only add the pure premium fields to operational loss for the time being. All the fields contain numeric floating-point values and we’d be adding seven different coverages and a total pure premium column to our table.

CTE for Breaking Down ULM Scores



Explanation: This query calculates the earned exposure of a row in Auto Operational Loss as a percentage and multiplies it by the ULM amounts based on the mapping shown above. The two tables involved are the ULM Auto view and our Auto Operational Loss dataset. For reference, the earned exposure column in our dataset is the DERIV\_ADJ\_EP\_EXPSR\_MO\_CNT column.

DML Query for Updating Auto Operational Loss



Explanation: This query populates the newly added columns based on the values derived from the CTE query. Our auto operational loss table is joined to the CTE on multiple attributes, including the policy ID, the policy period ID, the coverage type, and the earned month. This way, we can ensure that the correct pure premium value is being assigned to the correct row in our operational loss table. It is important to note that the ZEROIFNULL function is used because in the ULM view, the Policy Period ID field is a numeric column, and we are joining it to a varchar column in operational loss.

For any questions regarding this documentation and project, please reach out to:

|  |  |
| --- | --- |
| Peter Alonzo (Associate Data Engineer) | [peter.alonzo@thehartford.com](mailto:peter.alonzo@thehartford.com) |
| Kevin Bodie (Director of Data Engineering) | [kevin.bodie@thehartford.com](mailto:kevin.bodie@thehartford.com) |
| Karissa Christiano (Director of Product Analytics) | [karissa.christiano@thehartford.com](mailto:karissa.christiano@thehartford.com) |

SQL Terms:

**DDL (Data Definition Language):** Query that defines the database, schema, or table structure.

**DML (Data Manipulation Language):** Query that manipulates data within a database.

**CTE (Common Table Expression):** A temporary result set in SQL whose results can be referenced in another query.