IGDNT 3S3: How SmugMug increased data modeling productivity with Amazon Q Developer

by Geoff Ryder and Eddie Yao

March 13, 2025

Link to the samples github repo	https://github.com/zenbananas19/ai-for-data-modeling
Link to the related blog post	https://aws.amazon.com/blogs/devops/how-smugmug-increased-data-modeling-productivity-with-amazon-q-developer/

List of Prompt Examples

Prompt	Chat & Workspace Content	Response
@workspace What columns of database tables sample.orderinfo and sample.skuinfo would be best to left join skuinfo to orderinfo? Provide SQL code for the join. Draw an entity relationship diagram that shows the joins between the two tables, and includes only the fields involved in the join. Add a crow's foot cardinality marker to indicate a 1:many relationship, and add it next to the high cardinality table. Please create mermaid code to display this ERD.	DDL of sample.skuinfo, DDL of sample.orderinfo, and optionally a few rows of sample data.	

```
SELECT *
FROM dbt_geoff.orderinfo o
LEFT JOIN dbt_geoff.skuinfo s
ON o.sku_id = s.sku_id;

erDiagram
    orderinfo }o--|| skuinfo : "sku_id"
    orderinfo {
        double sku_id
    }
    skuinfo {
        double sku_id
    }
```

Please create graphviz code to display the ERD above.	Chat history up to this point.	

```
digraph ERD {
    rankdir=LR;
    node [shape=record];

    skuinfo [label="skuinfo|{sku_id: bigint}"];
    orderinfo [label="orderinfo|{sku_id: bigint}"];

    skuinfo -> orderinfo [arrowhead=crow, arrowtail=none, dir=both, label="1:N"];
}
```

Please create Python code to use in an IPython notebook that displays the ERD by calling https://mermaid.ink.	Chat history up to this point.	

```
import base64
from IPython.display import Image, display

def display_mermaid(graph):
    graphbytes = graph.encode("ascii")
    base64_bytes = base64.b64encode(graphbytes)
    base64_string = base64_bytes.decode("ascii")
    display(
        Image(
            url="https://mermaid.ink/img/" + base64_string
        )
    )

# Define the ERD diagram
mermaid_code = """erDiagram
    skuinfo ||--o{ orderinfo : "1:N"
    skuinfo {
        bigint sku_id
    }
    orderinfo {
        bigint sku_id
    }"""

# Display the diagram
display_mermaid(mermaid_code)
```

@workspace I am a Redshift database administrator and I am working on a data modeling problem. I would like to write SQL statements to left join table sample.skuinfo to sample.orderinfo. Please write that SQL to join the two tables. Also, I would like to write a SQL case statement to recover all string values defined in PHP that are represented as integer values in the database table.	PHP sample, and the chat history up to this point that has DDLs of the two tables.	

```
SELECT
    o.*,
    s.*,
    CASE s.vendor_id
        WHEN 1 THEN 'Vendor1'
        WHEN 2 THEN 'Vendor2'
        WHEN 3 THEN 'Vendor3'
        ELSE 'Unknown Vendor'
    END AS vendor_name
FROM sample.orderinfo o
LEFT JOIN sample.skuinfo s
    ON o.sku_id = s.sku_id;
```

Please write a short SQL test to compute counts of the key fields that are used in the joins, which will verify the cardinality assignments indicated in the entity relationship diagram above. The SQL test should compare distinct counts to total counts and null counts when it verifies the cardinality.	Chat history up to this point.	

```sql
WITH counts AS (
    -- Count analysis for orderinfo table
    SELECT
        'orderinfo' as table_name,
        COUNT(*) as total_rows,
        COUNT(sku_id) as non_null_count,
        COUNT(DISTINCT sku_id) as distinct_count,
        COUNT(*) - COUNT(sku_id) as null_count
    FROM sample.orderinfo

    UNION ALL

    -- Count analysis for skuinfo table
    SELECT
        'skuinfo' as table_name,
        COUNT(*) as total_rows,
        COUNT(sku_id) as non_null_count,
        COUNT(DISTINCT sku_id) as distinct_count,
        COUNT(*) - COUNT(sku_id) as null_count
    FROM sample.skuinfo
)
SELECT
    table_name,
    total_rows,
    non_null_count,
    distinct_count,
    null_count,
    CASE
        WHEN table_name = 'orderinfo' AND distinct_count <= (SELECT distinct_count FROM counts WHERE table_name = 'skuinfo')
        THEN 'Valid many-to-one relationship'
        WHEN table_name = 'skuinfo' AND distinct_count = non_null_count
        THEN 'Valid one side (unique values)'
        ELSE 'INVALID relationship'
    END as cardinality_check
FROM counts
ORDER BY table_name;
```

Please write a Python XGBoost program to predict order count by sku_description. Please also use the xgbfir library to identify feature importance, and rank the importance of 3-deep interactions among the predictor variables.	Chat history up to this point.	

```
# Please see the project repository for the full output.
# Some key parts of the response:

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error, r2_score
import xgboost as xgb
import xgbfir

# ... data preparation steps

# Initialize XGBoost model
model = xgb.XGBRegressor(
      objective='reg:squarederror',
      n_estimators=100,
      max_depth=3,
      learning_rate=0.1,
      random_state=42
    )

# ... model run steps

# 4. Analyze feature importance and interactions
def analyze_features(model, X):
    # Get feature names
    feature_names = X.columns.tolist()
    # Generate XGBFIR report
    xgbfir.saveXgbFI(
        model,
        feature_names=feature_names,
        OutputXlsxFile='feature_interactions.xlsx'
    )
    # Get feature importance
    importance_df = pd.DataFrame({
        'feature': feature_names,
        'importance': model.feature_importances_
    })
    importance_df = importance_df.sort_values('importance', ascending=False)
    return importance_df
```