Certainly, let's create a more advanced, detailed, and comprehensive workflow for predictive analytics using Python and deep analytics queries from a healthcare dataset. In this example, we'll focus on predicting patient readmission based on a range of features. This workflow includes data preprocessing, feature engineering, model building, hyperparameter tuning, and model evaluation.

Step 1: Advanced Hue SQL Queries for Data Extraction and Export to CSV

```
-- Query 1: Extract patient demographics, admission history, and diagnosis
SELECT
    P.PatientID,
    P.PatientName,
    P.Age,
    P.Gender,
    A.AdmissionDate,
    A.DischargeDate,
    A.Readmitted,
    M.MedicalDiagnosis
FROM
    DimensionPatient AS P
JOIN
    DimensionAdmissions AS A ON P.PatientID = A.PatientID
JOIN
    DimensionMedical AS M ON P.PatientID = M.PatientID
INTO OUTFILE '/tmp/patient_data.csv'
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';

-- Query 2: Extract medication history
SELECT
    P.PatientID,
    RX.MedicationName,
    RX.StartDate,
    RX.EndDate
FROM
    DimensionPrescriptions AS RX
JOIN
    DimensionPatient AS P ON RX.PatientID = P.PatientID
INTO OUTFILE '/tmp/medication_data.csv'
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

Step 2: Python Script for Data Preprocessing and Predictive Analytics

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.metrics import accuracy_score, classification_report

# Load the CSV files into Pandas DataFrames
patient_data = pd.read_csv('/tmp/patient_data.csv')
medication_data = pd.read_csv('/tmp/medication_data.csv')

# Merge patient, admission, and medication data
data = pd.merge(patient_data, medication_data, on='PatientID', how='inner')

# Data preprocessing and feature engineering
# Encode categorical variables
encoder = LabelEncoder()
data['Gender'] = encoder.fit_transform(data['Gender'])
data['MedicalDiagnosis'] = encoder.fit_transform(data['MedicalDiagnosis'])

# Calculate length of hospital stay
data['LengthOfStay'] = (pd.to_datetime(data['DischargeDate']) - pd.to_datetime(data['Admissi

# Extract features and target variable
X = data[['Age', 'Gender', 'MedicalDiagnosis', 'LengthOfStay']]
y = data['Readmitted']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model building and hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_rf_model = grid_search.best_estimator_

# Model evaluation
y_pred = best_rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Best Model Parameters: {grid_search.best_params_}')
print(f'Accuracy: {accuracy}')
```

```
print(f'Classification Report:\n{classification_rep}')
```

In this advanced workflow:

1. We extract and preprocess patient demographics, admission history, diagnosis, and medication data.
2. Categorical variables like 'Gender' and 'MedicalDiagnosis' are encoded using Label Encoding.
3. We engineer features such as 'LengthOfStay' as the length of hospital stay.
4. Data is split into training and testing sets.
5. We perform hyperparameter tuning using GridSearchCV to find the best Random Forest Classifier.
6. The model is evaluated using accuracy and a classification report.

This workflow is more detailed and comprehensive, covering data preprocessing, feature engineering, model selection, and evaluation. You can further enhance it by exploring additional features and advanced machine learning techniques based on your specific predictive analytics objectives.