Assignment Summary (Link)

&

Step By Step:(Link)

(Execution procedure and steps added clearly at the end of document)

1. Summary & Architecture

Project Objective: To engineer an end-to-end data pipeline that transforms raw telecom data into actionable churn risk intelligence. The solution moves beyond simple analysis to create a deployed inference engine capable of identifying at-risk customers with probabilistic confidence.

Architectural Overview:

The system follows a modern ELT + ML architecture:

Raw CSV --> Python ETL --> SQL Server (Star Schema) --> EDA --> Scikit-Learn Pipeline --> Inference Engine (.pkl)

2. Database Setup & Data Modeling

Design Philosophy: Before writing any code, we assessed the nature of the workload. Since the primary goal is trend analysis, churn metrics, and reporting, this necessitates an **OLAP (Online Analytical Processing)** approach rather than a transactional (OLTP) one.

To support high-performance querying and intuitive dashboarding, we implemented a **Star Schema**. This design surrounds a central Fact table with descriptive Dimension tables, optimizing the database for aggregations and minimizing join complexity.

Deliverable: HemoDataDB_Script.sql

Schema Objects:

- **dim_customer:** Stores demographic attributes (Gender, SeniorCitizen, Partner).
- **dim_service:** Stores subscription details (Internet, Phone, Monthly Charges).
- **fact_subscription:** The central transactional table containing keys to dimensions and quantitative metrics (Tenure, TotalCharges).

Implementation Details:

- **Referential Integrity:** Rigid Foreign Key constraints were defined to prevent orphaned records.
- **Indexing:** Primary keys were implemented on surrogate columns (customer_dim_id) rather than business keys to ensure database performance.

Database Setup and Model Diagram (PBIX Model view):

SQLQuery1.sql - MS...B (MSI\arunn (60))*

```
CREATE TABLE telco.dim_service (
    service_dim_id INT IDENTITY(1,1) PRIMARY KEY,
    phone_service BIT,
    multiple_lines VARCHAR(50),
    internet_service VARCHAR(50),
    online_security VARCHAR(50),
    online_backup VARCHAR(50),
    device_protection VARCHAR(50),
    tech_support VARCHAR(50),
    streaming_tv VARCHAR(50),
    streaming_movies VARCHAR(50),
    monthly_charges DECIMAL(10,2) -- Correctly placed in Dimension
);
GO

IF OBJECT_ID('telco.fact_subscription', 'U') IS NOT NULL DROP TABLE telco.fact_subscription;
CREATE TABLE telco.fact_subscription (
    fact_id INT IDENTITY(1,1) PRIMARY KEY,
    customer_dim_id INT,
```

90 %

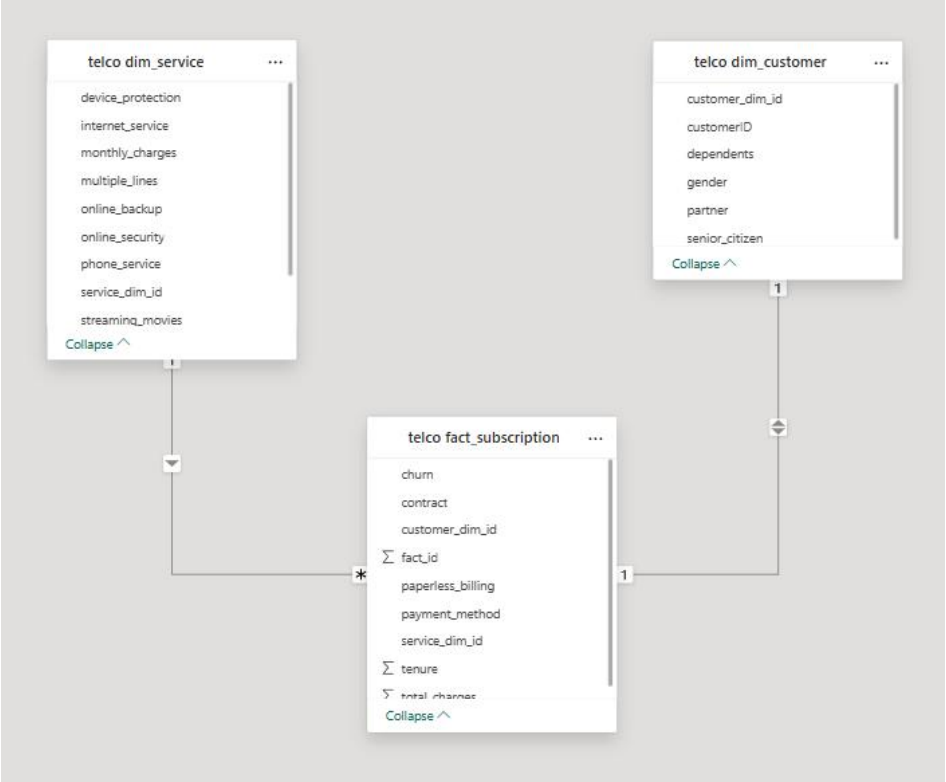
Results | Messages

	fact_id	customer_dim_id	service_dim_id	tenure	contract	paperless_billing	payment_method	total_charges	churn
1	1	1	1	1	Month-to-month	1	Electronic check	29.85	0
2	2	2	2	34	One year	0	Mailed check	1889.50	0
3	3	3	3	2	Month-to-month	1	Mailed check	108.15	1
4	4	4	4	45	One year	0	Bank transfer (automatic)	1840.75	0
5	5	5	5	2	Month-to-month	1	Electronic check	151.65	1
6	6	6	6	8	Month-to-month	1	Electronic check	820.50	1
7	7	7	7	22	Month-to-month	1	Credit card (automatic)	1949.40	0
8	8	8	8	10	Month-to-month	0	Mailed check	301.90	0

	service_dim_id	phone_service	multiple_lines	internet_service	online_security	online_backup	device_protection	tech_support	streaming_tv	streaming_movies	monthly_charges
1	1	0	No phone service	DSL	No	Yes	No	No	No	No	29.85
2	2	1	No	DSL	Yes	No	Yes	No	No	No	56.95
3	3	1	No	DSL	Yes	Yes	No	No	No	No	53.85
4	4	0	No phone service	DSL	Yes	No	Yes	Yes	No	No	42.30
5	5	1	No	Fiber optic	No	No	No	No	No	No	70.70
6	6	1	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	99.65
7	7	1	Yes	Fiber optic	No	Yes	No	No	Yes	No	89.10
8	8	0	No phone service	DSL	Yes	No	No	No	No	No	29.75

	customer_dim_id	customerID	gender	senior_citizen	partner	dependents
1	1	7590-VHVEG	Female	0	1	0
2	2	5575-GNVDE	Male	0	0	0
3	3	3668-QPYBK	Male	0	0	0
4	4	7795-CFOCW	Male	0	0	0
5	5	9237-HQITU	Female	0	0	0
6	6	9305-CDSKC	Female	0	0	0
7	7	1452-KIOVK	Male	0	0	1

Query executed successfully. MSI\SQLE



3. ETL Process (Extract, Transform, Load)

Script: HemoDataTest_ETL.py

Engineering Rationale:

Instead of a simple "bulk load," I built an Object-Oriented ETL Class (TelcoETL). This approach allows for modularity, better error handling, and easier testing compared to linear procedural scripts.

1. Extract:

- Ingests raw CSV data using Pandas.

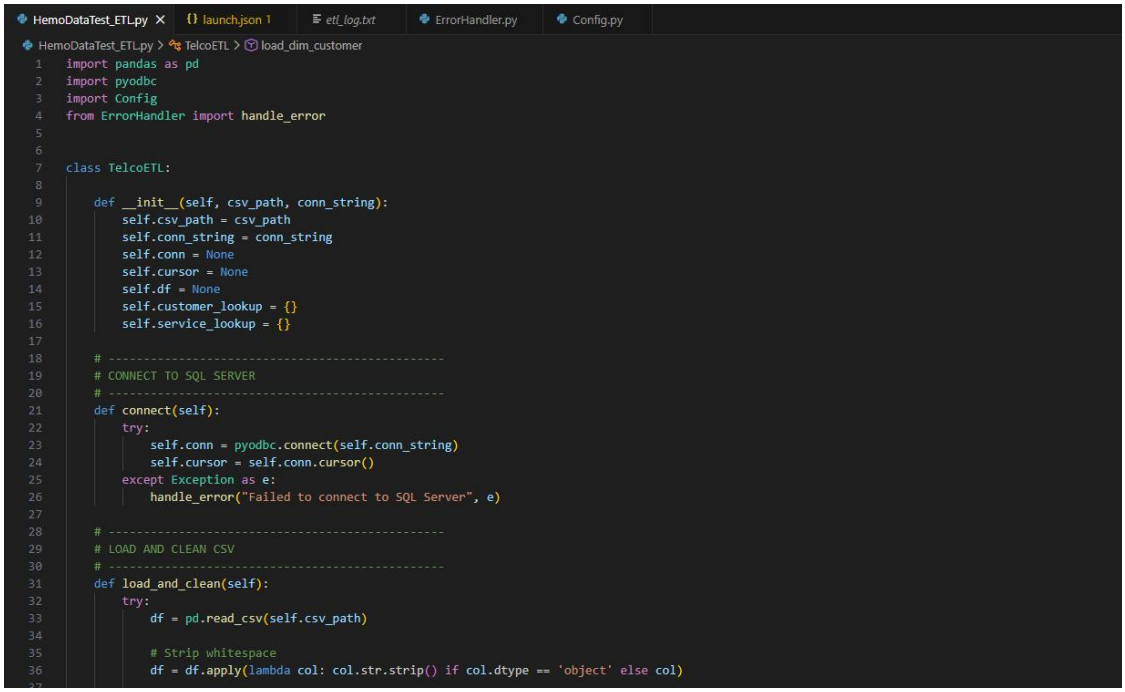
2. Transform:

- **Type Casting:** Converted categorical "Yes/No" string fields into Binary integers (1/0).
- **Data Sanitization:** Detected empty strings in the TotalCharges column (representing new customers with 0 tenure) and imputed them with the median.

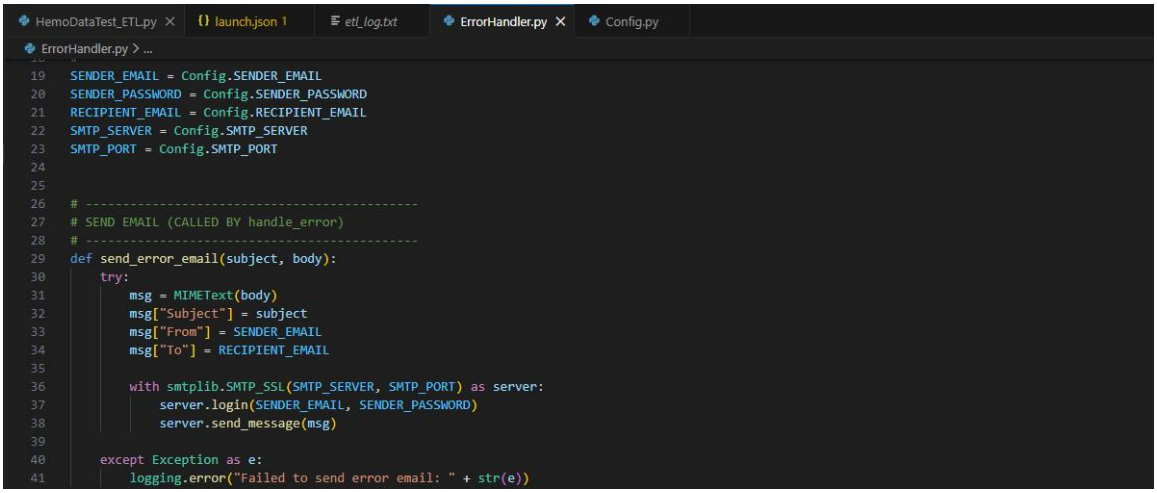
3. Load:

- The script first populates the Dimension tables, retrieves the newly generated Surrogate Keys, and then maps these keys to the Fact table.
- *Why?* This maintains the Star Schema integrity. If any part of the transaction fails, the entire batch rolls back, preventing data corruption.

1) **ETL Pipeline** (Object Oriented - All secrets in config and error handler for run history & alerts)

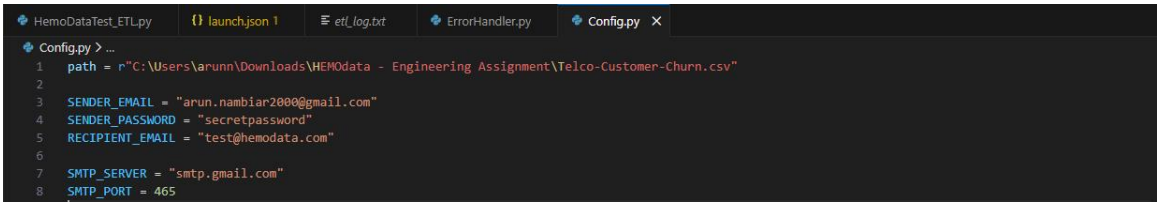


2) **Error Handler** (Traces error messages using logger package in txt file and also sends email alert)



```python
SENDER_EMAIL = Config.SENDER_EMAIL
SENDER_PASSWORD = Config.SENDER_PASSWORD
RECIPIENT_EMAIL = Config.RECIPIENT_EMAIL
SMTP_SERVER = Config.SMTP_SERVER
SMTP_PORT = Config.SMTP_PORT


# ------------------------------------------
# SEND EMAIL (CALLED BY handle_error)
# ------------------------------------------
def send_error_email(subject, body):
    try:
        msg = MIMEText(body)
        msg["Subject"] = subject
        msg["From"] = SENDER_EMAIL
        msg["To"] = RECIPIENT_EMAIL

        with smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT) as server:
            server.login(SENDER_EMAIL, SENDER_PASSWORD)
            server.send_message(msg)

    except Exception as e:
        logging.error("Failed to send error email: " + str(e))
```

3) **Config File** (has all paths, secrets and any sensitive information such as email addresses)

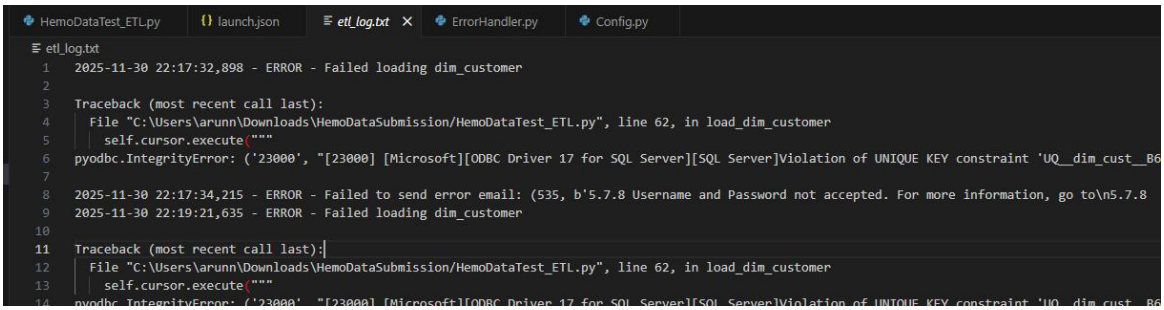


```python
path = r"C:\Users\arunn\Downloads\HEMOdata - Engineering Assignment\Telco-Customer-Churn.csv"

SENDER_EMAIL = "arun.nambiar2000@gmail.com"
SENDER_PASSWORD = "secretpassword"
RECIPIENT_EMAIL = "test@hemodata.com"

SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 465
```

4) **Launch.json** (as we have multiple files referring each other we need a JSON file defining flow)

```json
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Run HemoData ETL",
            "type": "python",
            "request": "launch",
            "program": "${workspaceFolder}/HemoDataTest_ETL.py",
            "console": "integratedTerminal"
```

5) **Error Handler TXT file** (keeps logs of errors in the executions -- keeps appending)



```
2025-11-30 22:17:32,898 - ERROR - Failed loading dim_customer

Traceback (most recent call last):
  File "C:\Users\arunn\Downloads\HemoDataSubmission/HemoDataTest_ETL.py", line 62, in load_dim_customer
    self.cursor.execute("""
pyodbc.IntegrityError: ('23000', "[23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Violation of UNIQUE KEY constraint 'UQ__dim_cust__B6

2025-11-30 22:17:34,215 - ERROR - Failed to send error email: (535, b'5.7.8 Username and Password not accepted. For more information, go to\n5.7.8
2025-11-30 22:19:21,635 - ERROR - Failed loading dim_customer

Traceback (most recent call last):
  File "C:\Users\arunn\Downloads\HemoDataSubmission/HemoDataTest_ETL.py", line 62, in load_dim_customer
    self.cursor.execute("""
pyodbc.IntegrityError: ('23000', "[23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Violation of UNIQUE KEY constraint 'UQ__dim_cust__B6
```

4. Exploratory Data Analysis (EDA)

Script: EDA.py

Engineering Rationale: The goal of EDA was not just to "make charts," but to perform **Feature Selection** and validate **Data Quality** before training expensive models.

Instead of manually writing boilerplate code for histograms and correlation matrices, I leveraged **ydata-profiling,** an industry-standard automated analysis tool.

4.1 Automated Data Quality Assessment

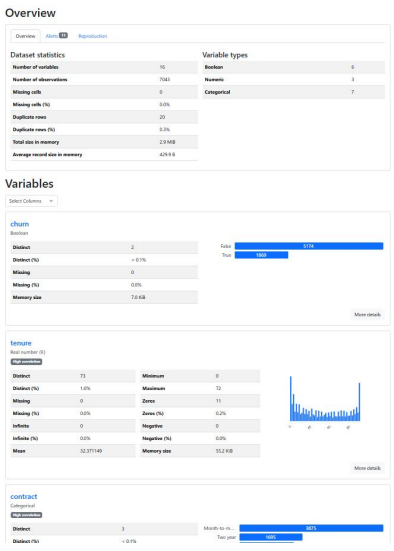
Before modeling, ensuring data integrity is critical. The profiling engine automatically audited the dataset for:

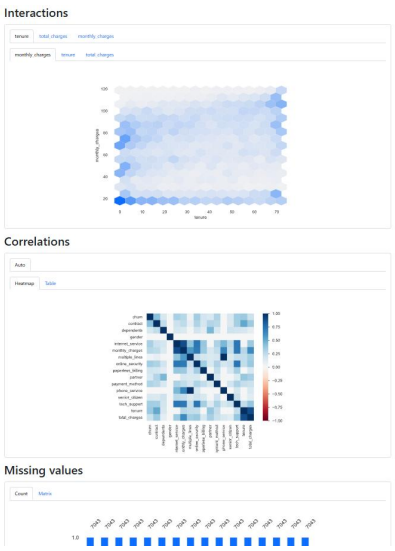
- **Missing Data:** Visualized the sparsity of the dataset to decide between dropping rows or imputing values (e.g., handling the TotalCharges gaps).
- **Cardinality:** identified columns with constant values (zero variance) or high cardinality that would add noise to the machine learning model.
- **Duplicate Detection:** Ensured that no customer records were duplicated, preventing bias in the training phase.

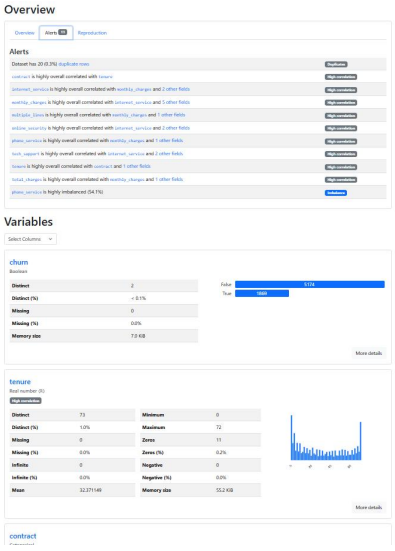
4.2 Pattern Recognition & Correlation

The tool generated a comprehensive **HTML** report mapping fields against each other to uncover hidden relationships: (kindly open HTML File once running EDA file it should be in your folder)

- **Correlation Analysis:** It calculated Pearson and Spearman correlation matrices automatically. This allowed me to detect patterns (e.g., Tenure vs. TotalCharges) and decide which redundant features to drop to improve model performance.
- **Interaction Mapping:** It visualized interactions between key variables, such as how Contract Type interacts with Monthly Charges to drive Churn.







5. Machine Learning Pipeline

Scripts: 1_compare_models.py (Training) & 2_predict_auto.py (Inference)

5.1 The "Model Tournament"

Instead of arbitrarily choosing an algorithm, I implemented a competitive selection process:

Benchmarking: I trained six distinct algorithms (Logistic Regression, Random Forest, Gradient Boosting, SVM, MLP, KNN) on the same split of data.

Metric Selection: I prioritized **F1-Score** over Accuracy.

Why? Churn datasets are imbalanced (73% Stay / 27% Churn). A model that predicts "Stay" for everyone would have 73% accuracy but zero business value. F1-Score balances Precision and Recall to ensure we catch actual churners.

5.2 Selected Architecture: Logistic Regression

Winner: Logistic Regression

- **Class Weighting:** I utilized class_weight='balanced' to automatically adjust the penalty for misclassifying the minority class (Churners), ensuring the model prioritizes recall over simple accuracy.

5.3 Inference

Serialization: The winning pipeline was serialized into a .pkl (Pickle) artifact.

This second script takes input from the user forthings like employee id, age, gender etc and predicts the probability of churn using the model with the best outcome.

THE REASON FOR keeping model selection and pkl file generation as dynamic was that as a company would keep adding data we would have a better sample space and some models that might have not performed well in such a small data would shine such as random forest. So ever time you run the ML script, new pkl file gets generated according to the best F score and its used for prediction.

Output: The inference script outputs a **Probability Score** (e.g., 85%), not just a binary prediction.

Business Value: This allows the marketing team to segment customers into "High Risk" vs. "Medium Risk" tiers, optimizing their retention budget.

1) ML Model Training and testing for F score/ best model determination.

```
Training 11 models... this might take a minute.

================================================================
Model Name           | F1 Score   | ROC-AUC    | Accuracy
================================================================
LogisticRegression   | 0.6179     | 0.8359     | 0.7367
DecisionTree         | 0.6048     | 0.8278     | 0.7339
RandomForest         | 0.5278     | 0.8112     | 0.7828
GradientBoosting     | 0.5732     | 0.8406     | 0.7991
HistGradientBoost    | 0.5356     | 0.8273     | 0.7821
AdaBoost             | 0.5920     | 0.8387     | 0.8034
SVM (Linear)         | 0.5855     | 0.8294     | 0.6884
SVM (RBF)            | 0.6067     | 0.8104     | 0.7331
KNN                  | 0.5068     | 0.7507     | 0.7445
NaiveBayes           | 0.5727     | 0.8039     | 0.6686
NeuralNet (MLP)      | 0.4560     | 0.7462     | 0.7410
----------------------------------------------------------------
The winner is LogisticRegression with F1 Score: 0.6179
SAVED: Only the best model was saved to 'model_LogisticRegression.pkl'.
PS C:\Users\arunn\Downloads\HemoData_ML_Submission> & C:/Users/arunn/AppData/Local/Programs/Python/Python312/python.exe c:
```

2) After the best model is exported into a pkl (pickle) file (sort of like the brain of an ml model trained and saved model) we use it to input data and predict outcomes such as here I have inputted a new customers data and it tells us the probability (by using a package in sci kit learn that lets us calculate the probability of outcome using the accuracy of the model and input data) of the new customer to be churned.

Here we can see that the customer is 32.8% at risk of being churned.

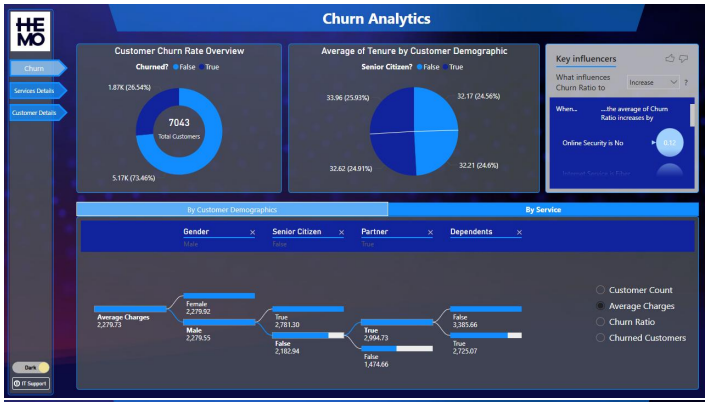
```
--- Select Options (Type the name) ---
Payment Methods: Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic)
Payment Method: Mailed check
Contract Types: Month-to-month, One year, Two year
Contract: One year
Internet Service: DSL, Fiber optic, No
Internet Service: DSL
Gender (Male/Female): Male

--- Quick Yes/No Questions ---
Senior Citizen? (Yes/No or 1/0): 1
Has Partner? (Yes/No or 1/0): 2
Please type 'Yes' or 'No'.
Has Partner? (Yes/No or 1/0): 1
Has Dependents? (Yes/No or 1/0):
Please type 'Yes' or 'No'.
Has Dependents? (Yes/No or 1/0): 1
Has Phone Service? (Yes/No or 1/0): 1
Paperless Billing? (Yes/No or 1/0): 0

Running prediction using model_LogisticRegression.pkl...

==================================================
FINAL PREDICTION RESULT
==================================================
STATUS: SAFE (ACTIVE)
Probability of Leaving: 32.8%
Recommendation: Customer is stable.
==================================================
```

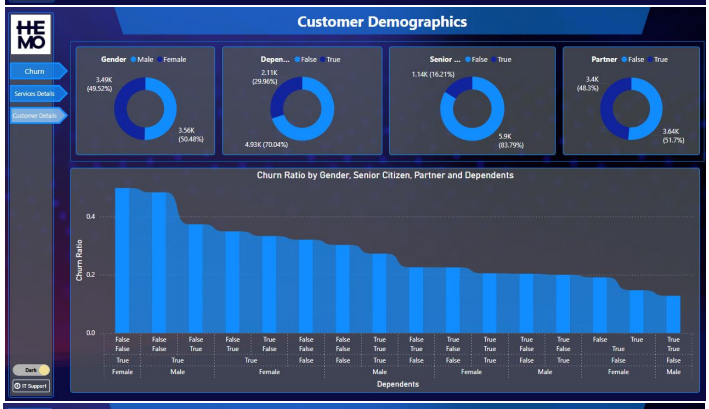
6. Data Visualization (Power BI)

Deliverable: HemoData_Telco_ChurnAnalytics.pbix

To bridge the gap between technical data science and business strategy, I deployed an interactive dashboard.

- **Key Influencers Visual:** Leveraging Power BI's AI capabilities to automatically surface segments with high churn rates.
- **Decomposition Tree:** Enabling stakeholders to perform "Root Cause Analysis" by drilling down into specific demographics or service categories dynamically.
- **Parameterization:** Users can slice data by Service or Customer attributes to simulate different business scenarios.





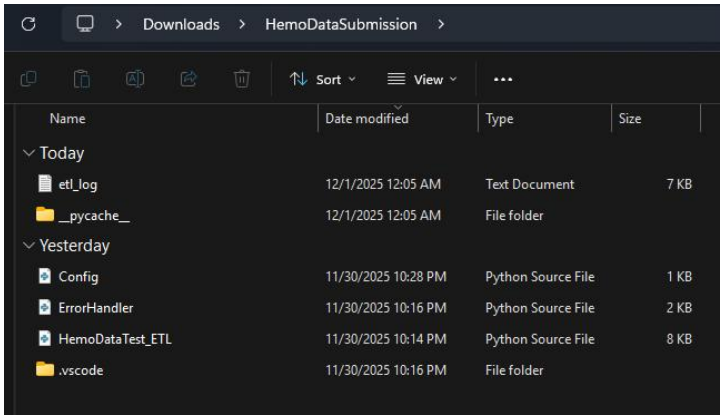
IMPORTANT

7. Step By Step Execution Procedure

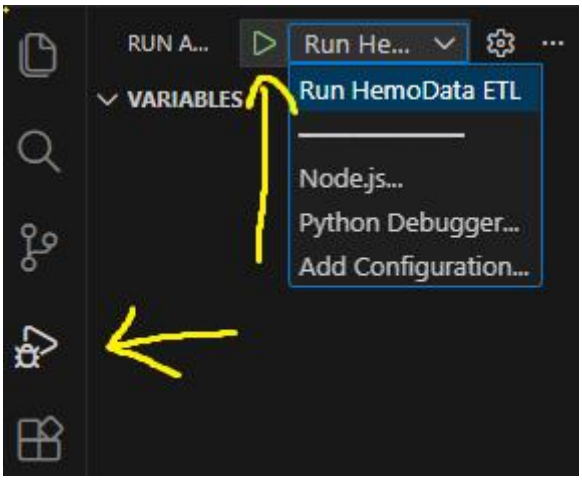
ALL FILES CAN BE FOUND IN FOLDERS IN THIS LINK IF YOU WANT TO SKIP ANY SETUP STEPS(Link)

1) Run the SQL Script (HemoData_Arun_DB Script)
2) Once DB is successully created --> Open Visual Studio Code and load the below listed files in one folder (please change ALL file paths accordingly in config) and open the folder in Visual studio. (pip install pandas pyodbc)

- HemoDataTest_ETL.py
- ErrorHandler.py
- Config.py

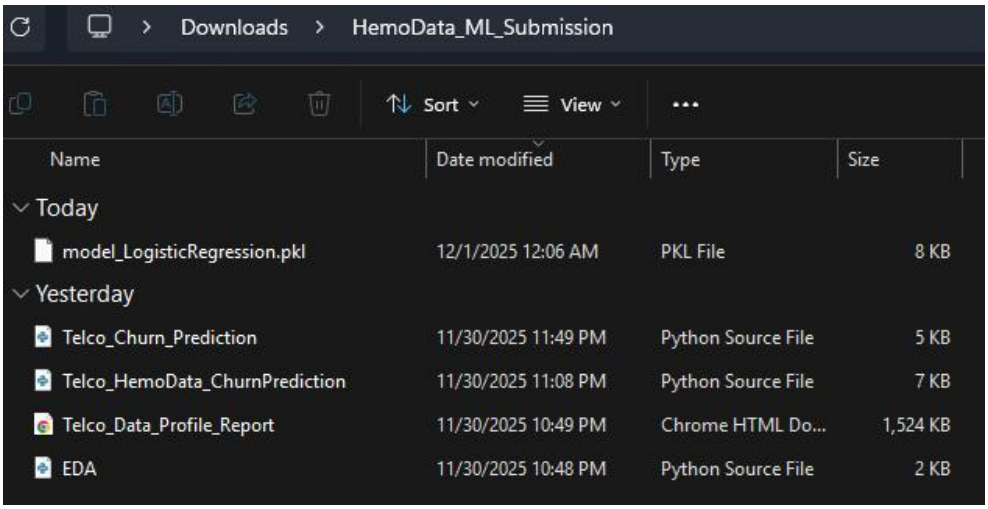


Create a .vscode folder and put the launch.json file in it.Then open VS Code and go to debug to run the launch json config (RUN HEMODATA ETL)

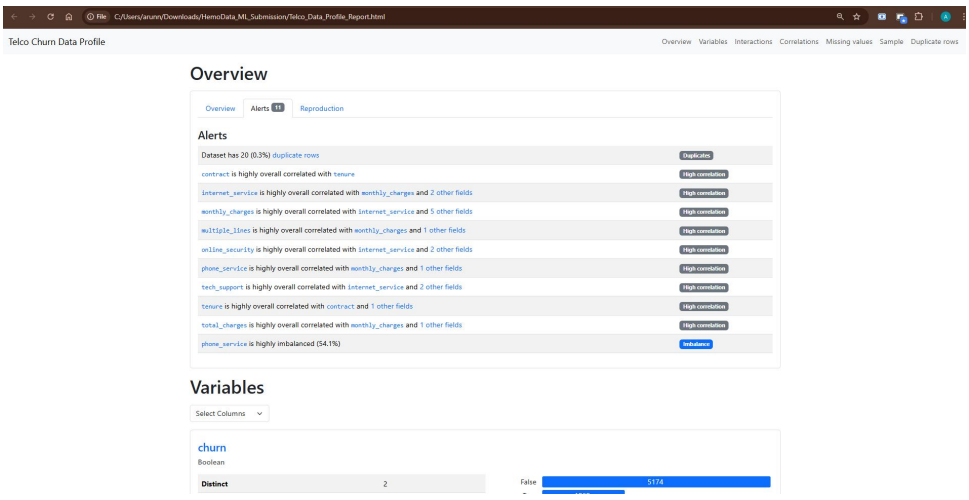


ETL DONE!

3) Now for ML you can run all files individually as they are not correlated, but since pkl file needs to be created and fetched in the prediction py file recommended is that you create a folder for ml and put all files in it.



4) Kindly run EDA File, and once done a HTML file will be generated please open it for detailed EDA which looks like the following: (pip install ydata-profiling seaborn matplotlib setuptools)



5) Run ML File to create PKL file (if you want to skip use the pkl file I have in the github link at the beginning).(pip install ydata-profiling seaborn matplotlib setuptools)

6) Run the prediction file, give inputs and get the predicted churn probability.