DATA WAREHOUSE MINING (CT-463)

Department of Computer Science and Information Technology

PROJECT: DATA WAREHOUSE DESIGN IMPLEMENTATION

GROUP MEMBERS:

1. Muhammad Abdul Rafay Shaikh (CT-21035).
2. Rohan Ahmed Siddiqui (CT-21050).
3. Muhammad Umair Shaikh (CT-21048).
4. Syed Sumam Zaidi (CT-21038).

Section: A.

Contents

DATA FROM CSV

Introduction

The objective of this dataset is to analyze data collected from a survey (likely on employment, educational level, coding experience, and salary) and develop a machine learning model to predict annual salary. The dataset, structured with variables like Country, Education Level (EdLevel), Years of Professional Coding Experience (YearsCodePro), Employment, and Annual Compensation (ConvertedCompYearly), is explored and refined to extract insights and prepare it for predictive modeling.

Data Extraction:

The initial data extraction step involves reading the survey data into a pandas DataFrame. The data is loaded from a CSV file containing various columns, with only the relevant ones—Country, EdLevel, YearsCodePro, Employment, and ConvertedCompYearly—being retained for analysis.

```
df.head()
```

	ResponseId	MainBranch	Age	Employment	RemoteWork	Check	CodingActivities	EdLevel	LearnCode	LearnCodeOnline	...	JobSatPoints_6	JobSatPoints_7	JobSatPoints_8	JobSatPoints_9	JobSatPoints_10	JobSatPoints_1
0	1	I am a developer by profession	Under 18 years old	Employed, full-time	Remote	Apples	Hobby	Primary/elementary school	Books / Physical media	NaN	...	NaN	NaN	NaN	NaN	NaN	Na
1	2	I am a developer by profession	35-44 years old	Employed, full-time	Remote	Apples	Hobby;Contribute to open-source projects;Other...	Bachelor's degree (B.A., B.S., B.Eng., etc.)	Books / Physical media;Colleague;On the job tr...	Technical documentation;Blogs;Books;Written Tu...	...	0.0	0.0	0.0	0.0	0.0	0.
2	3	I am a developer by profession	45-54 years old	Employed, full-time	Remote	Apples	Hobby;Contribute to open-source projects;Other...	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	Books / Physical media;Colleague;On the job tr...	Technical documentation;Blogs;Books;Written Tu...	...	NaN	NaN	NaN	NaN	NaN	Na
3	4	I am learning to code	18-24 years old	Student, full-time	NaN	Apples	NaN	Some college/university study without earning ...	Other online resources (e.g., videos, blogs, f...	Stack Overflow;How-to videos;Interactive tutorial	...	NaN	NaN	NaN	NaN	NaN	Na
4	5	I am a developer by profession	18-24 years old	Student, full-time	NaN	Apples	NaN	Secondary school (e.g. American high school, G...	Other online resources (e.g., videos, blogs, f...	Technical documentation;Blogs;Written Tutorial...	...	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 114 columns

Data Transformation:

The dataset undergoes a series of transformations to ensure it is suitable for modeling. This process includes:

- **Data Cleaning**: Handling missing values, if any, in critical columns, and filtering out or imputing invalid or outlier entries.
- **Feature Engineering**: Transforming `YearsCodePro` (years of professional experience) and `EdLevel` (educational level) into numerical forms if necessary. These variables may also be normalized or categorized to improve the model's accuracy.
- **Encoding Categorical Variables**: Variables like `Country` and `Employment` are converted into numerical values through encoding methods (e.g., one-hot encoding or label encoding).
- **Data Splitting**: The data is split into training and test sets to allow for model training and evaluation.

	Country	EdLevel	YearsCodePro	Employment	Salary
72	Pakistan	Secondary school (e.g. American high school, G...	1	Employed, full-time;Student, full-time;Indepen...	7322.0
374	Austria	Professional degree (JD, MD, Ph.D, Ed.D, etc.)	6	Employed, full-time	30074.0
379	Turkey	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	6	Employed, full-time	91295.0
385	France	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	17	Independent contractor, freelancer, or self-em...	53703.0
389	United States of America	Some college/university study without earning ...	7	Employed, full-time;Student, part-time	110000.0

Data Load:

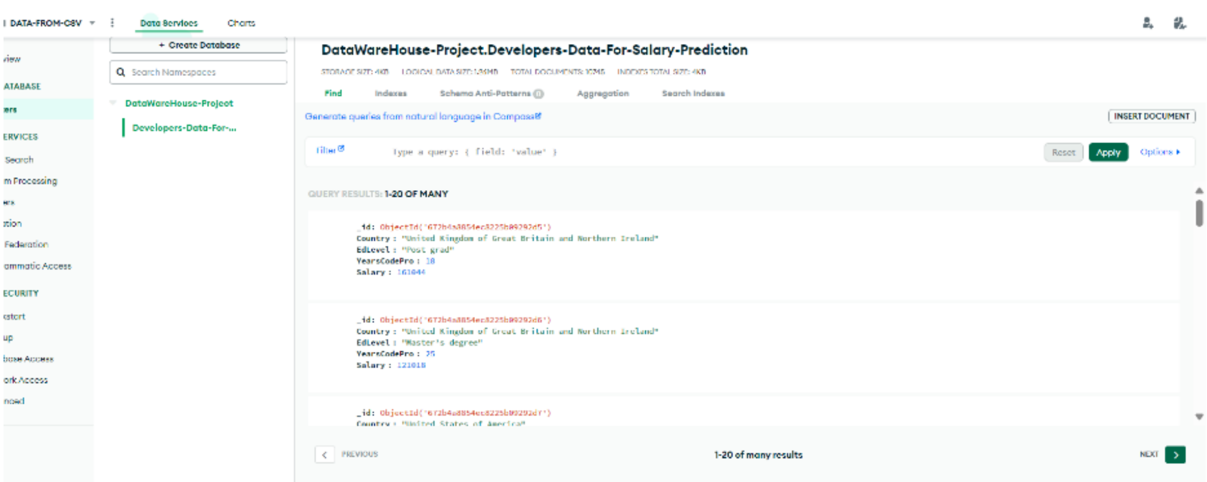
Once transformed, the prepared data is loaded into the machine learning pipeline, where it undergoes further processing, including scaling and any necessary adjustments to optimize the data for model training.

```
data_dict = transformed_df.to_dict("records")  # Convert DataFrame to a list of dictionaries
collection.insert_many(data_dict)
```


```
InsertManyResult([ObjectId('672b4a8854ec8225b09292d5'), ObjectId('672b4a8854ec8225b09292d6'), ObjectId('672b4a8854ec8225b09292d7'), ObjectId('672b4a8854ec8225b09292d8'), ObjectId('672b4a8854ec8225b09292d9'), ObjectId('672b4a8854ec8225b0
ObjectId('672b4a8854ec8225b09292db'), ObjectId('672b4a8854ec8225b09292dc'), ObjectId('672b4a8854ec8225b09292dd'), ObjectId('672b4a8854ec8225b09292de'), ObjectId('672b4a8854ec8225b09292df'), ObjectId('672b4a8854ec8225b09292e0'),
ObjectId('672b4a8854ec8225b09292e1'), ObjectId('672b4a8854ec8225b09292e2'), ObjectId('672b4a8854ec8225b09292e3'), ObjectId('672b4a8854ec8225b09292e4'), ObjectId('672b4a8854ec8225b09292e5'), ObjectId('672b4a8854ec8225b09292e6'),
ObjectId('672b4a8854ec8225b09292e7'), ObjectId('672b4a8854ec8225b09292e8'), ObjectId('672b4a8854ec8225b09292e9'), ObjectId('672b4a8854ec8225b09292ea'), ObjectId('672b4a8854ec8225b09292eb'), ObjectId('672b4a8854ec8225b09292ec'),
ObjectId('672b4a8854ec8225b09292ed'), ObjectId('672b4a8854ec8225b09292ee'), ObjectId('672b4a8854ec8225b09292ef'), ObjectId('672b4a8854ec8225b09292f0'), ObjectId('672b4a8854ec8225b09292f1'), ObjectId('672b4a8854ec8225b09292f2'),
ObjectId('672b4a8854ec8225b09292f3'), ObjectId('672b4a8854ec8225b09292f4'), ObjectId('672b4a8854ec8225b09292f5'), ObjectId('672b4a8854ec8225b09292f6'), ObjectId('672b4a8854ec8225b09292f7'), ObjectId('672b4a8854ec8225b09292f8'),
ObjectId('672b4a8854ec8225b09292f9'), ObjectId('672b4a8854ec8225b09292fa'), ObjectId('672b4a8854ec8225b09292fb'), ObjectId('672b4a8854ec8225b09292fc'), ObjectId('672b4a8854ec8225b09292fd'), ObjectId('672b4a8854ec8225b09292fe'),
ObjectId('672b4a8854ec8225b09292ff'), ObjectId('672b4a8854ec8225b0929300'), ObjectId('672b4a8854ec8225b0929301'), ObjectId('672b4a8854ec8225b0929302'), ObjectId('672b4a8854ec8225b0929303'), ObjectId('672b4a8854ec8225b0929304'),
ObjectId('672b4a8854ec8225b0929305'), ObjectId('672b4a8854ec8225b0929306'), ObjectId('672b4a8854ec8225b0929307'), ObjectId('672b4a8854ec8225b0929308'), ObjectId('672b4a8854ec8225b0929309'), ObjectId('672b4a8854ec8225b092930a'),
ObjectId('672b4a8854ec8225b092930b'), ObjectId('672b4a8854ec8225b092930c'), ObjectId('672b4a8854ec8225b092930d'), ObjectId('672b4a8854ec8225b092930e'), ObjectId('672b4a8854ec8225b092930f'), ObjectId('672b4a8854ec8225b0929310'),
ObjectId('672b4a8854ec8225b0929311'), ObjectId('672b4a8854ec8225b0929312'), ObjectId('672b4a8854ec8225b0929313'), ObjectId('672b4a8854ec8225b0929314'), ObjectId('672b4a8854ec8225b0929315'), ObjectId('672b4a8854ec8225b0929316'),
ObjectId('672b4a8854ec8225b0929317'), ObjectId('672b4a8854ec8225b0929318'), ObjectId('672b4a8854ec8225b0929319'), ObjectId('672b4a8854ec8225b092931a'), ObjectId('672b4a8854ec8225b092931b'), ObjectId('672b4a8854ec8225b092931c'),
ObjectId('672b4a8854ec8225b092931d'), ObjectId('672b4a8854ec8225b092931e'), ObjectId('672b4a8854ec8225b092931f'), ObjectId('672b4a8854ec8225b0929320'), ObjectId('672b4a8854ec8225b0929321'), ObjectId('672b4a8854ec8225b0929322'),
ObjectId('672b4a8854ec8225b0929323'), ObjectId('672b4a8854ec8225b0929324'), ObjectId('672b4a8854ec8225b0929325'), ObjectId('672b4a8854ec8225b0929326'), ObjectId('672b4a8854ec8225b0929327'), ObjectId('672b4a8854ec8225b0929328'),
ObjectId('672b4a8854ec8225b0929329'), ObjectId('672b4a8854ec8225b092932a'), ObjectId('672b4a8854ec8225b092932b'), ObjectId('672b4a8854ec8225b092932c'), ObjectId('672b4a8854ec8225b092932d'), ObjectId('672b4a8854ec8225b092932e'),
ObjectId('672b4a8854ec8225b092932f'), ObjectId('672b4a8854ec8225b0929330'), ObjectId('672b4a8854ec8225b0929331'), ObjectId('672b4a8854ec8225b0929332'), ObjectId('672b4a8854ec8225b0929333'), ObjectId('672b4a8854ec8225b0929334'),
ObjectId('672b4a8854ec8225b0929335'), ObjectId('672b4a8854ec8225b0929336'), ObjectId('672b4a8854ec8225b0929337'), ObjectId('672b4a8854ec8225b0929338'), ObjectId('672b4a8854ec8225b0929339'), ObjectId('672b4a8854ec8225b092933a'),
ObjectId('672b4a8854ec8225b092933b'), ObjectId('672b4a8854ec8225b092933c'), ObjectId('672b4a8854ec8225b092933d'), ObjectId('672b4a8854ec8225b092933e'), ObjectId('672b4a8854ec8225b092933f'), ObjectId('672b4a8854ec8225b0929340'),
ObjectId('672b4a8854ec8225b0929341'), ObjectId('672b4a8854ec8225b0929342'), ObjectId('672b4a8854ec8225b0929343'), ObjectId('672b4a8854ec8225b0929344'), ObjectId('672b4a8854ec8225b0929345'), ObjectId('672b4a8854ec8225b0929346'),
ObjectId('672b4a8854ec8225b0929347'), ObjectId('672b4a8854ec8225b0929348'), ObjectId('672b4a8854ec8225b0929349'), ObjectId('672b4a8854ec8225b092934a'), ObjectId('672b4a8854ec8225b092934b'), ObjectId('672b4a8854ec8225b092934c'),
ObjectId('672b4a8854ec8225b092934d'), ObjectId('672b4a8854ec8225b092934e'), ObjectId('672b4a8854ec8225b092934f'), ObjectId('672b4a8854ec8225b0929350'), ObjectId('672b4a8854ec8225b0929351'), ObjectId('672b4a8854ec8225b0929352'),
ObjectId('672b4a8854ec8225b0929353'), ObjectId('672b4a8854ec8225b0929354'), ObjectId('672b4a8854ec8225b0929355'), ObjectId('672b4a8854ec8225b0929356'), ObjectId('672b4a8854ec8225b0929357'), ObjectId('672b4a8854ec8225b0929358'),
ObjectId('672b4a8854ec8225b0929359'), ObjectId('672b4a8854ec8225b092935a'), ObjectId('672b4a8854ec8225b092935b'), ObjectId('672b4a8854ec8225b092935c'), ObjectId('672b4a8854ec8225b092935d'), ObjectId('672b4a8854ec8225b092935e'),
ObjectId('672b4a8854ec8225b092935f'), ObjectId('672b4a8854ec8225b0929360'), ObjectId('672b4a8854ec8225b0929361'), ObjectId('672b4a8854ec8225b0929362'), ObjectId('672b4a8854ec8225b0929363'), ObjectId('672b4a8854ec8225b0929364'),
ObjectId('672b4a8854ec8225b0929365'), ObjectId('672b4a8854ec8225b0929366'), ObjectId('672b4a8854ec8225b0929367'), ObjectId('672b4a8854ec8225b0929368'), ObjectId('672b4a8854ec8225b0929369'), ObjectId('672b4a8854ec8225b092936a'),
ObjectId('672b4a8854ec8225b092936b'), ObjectId('672b4a8854ec8225b092936c'), ObjectId('672b4a8854ec8225b092936d'), ObjectId('672b4a8854ec8225b092936e'), ObjectId('672b4a8854ec8225b092936f'), ObjectId('672b4a8854ec8225b0929370'),
ObjectId('672b4a8854ec8225b0929371'), ObjectId('672b4a8854ec8225b0929372'), ObjectId('672b4a8854ec8225b0929373'), ObjectId('672b4a8854ec8225b0929374'), ObjectId('672b4a8854ec8225b0929375'), ObjectId('672b4a8854ec8225b0929376'),
ObjectId('672b4a8854ec8225b0929377'), ObjectId('672b4a8854ec8225b0929378'), ObjectId('672b4a8854ec8225b0929379'), ObjectId('672b4a8854ec8225b092937a'), ObjectId('672b4a8854ec8225b092937b'), ObjectId('672b4a8854ec8225b092937c'),
ObjectId('672b4a8854ec8225b092937d'), ObjectId('672b4a8854ec8225b092937e'), ObjectId('672b4a8854ec8225b092937f'), ObjectId('672b4a8854ec8225b0929380'), ObjectId('672b4a8854ec8225b0929381'), ObjectId('672b4a8854ec8225b0929382'),
ObjectId('672b4a8854ec8225b0929383'), ObjectId('672b4a8854ec8225b0929384'), ObjectId('672b4a8854ec8225b0929385'), ObjectId('672b4a8854ec8225b0929386'), ObjectId('672b4a8854ec8225b0929387'), ObjectId('672b4a8854ec8225b0929388'),
ObjectId('672b4a8854ec8225b0929389'), ObjectId('672b4a8854ec8225b092938a'), ObjectId('672b4a8854ec8225b092938b'), ObjectId('672b4a8854ec8225b092938c'), ObjectId('672b4a8854ec8225b092938d'), ObjectId('672b4a8854ec8225b092938e'),
ObjectId('672b4a8854ec8225b092938f'), ObjectId('672b4a8854ec8225b0929390'), ObjectId('672b4a8854ec8225b0929391'), ObjectId('672b4a8854ec8225b0929392'), ObjectId('672b4a8854ec8225b0929393'), ObjectId('672b4a8854ec8225b0929394'),
ObjectId('672b4a8854ec8225b0929395'), ObjectId('672b4a8854ec8225b0929396'), ObjectId('672b4a8854ec8225b0929397'), ObjectId('672b4a8854ec8225b0929398'), ObjectId('672b4a8854ec8225b0929399'), ObjectId('672b4a8854ec8225b092939a'),
ObjectId('672b4a8854ec8225b092939b'), ObjectId('672b4a8854ec8225b092939c'), ObjectId('672b4a8854ec8225b092939d'), ObjectId('672b4a8854ec8225b092939e'), ObjectId('672b4a8854ec8225b092939f'), ObjectId('672b4a8854ec8225b09293a0'),
ObjectId('672b4a8854ec8225b09293a1'), ObjectId('672b4a8854ec8225b09293a2'), ObjectId('672b4a8854ec8225b09293a3'), ObjectId('672b4a8854ec8225b09293a4'), ObjectId('672b4a8854ec8225b09293a5'), ObjectId('672b4a8854ec8225b09293a6'),
ObjectId('672b4a8854ec8225b09293a7'), ObjectId('672b4a8854ec8225b09293a8'), ObjectId('672b4a8854ec8225b09293a9'), ObjectId('672b4a8854ec8225b09293aa'), ObjectId('672b4a8854ec8225b09293ab'), ObjectId('672b4a8854ec8225b09293ac'),
ObjectId('672b4a8854ec8225b09293ad'), ObjectId('672b4a8854ec8225b09293ae'), ObjectId('672b4a8854ec8225b09293af'), ObjectId('672b4a8854ec8225b09293b0'), ObjectId('672b4a8854ec8225b09293b1'), ObjectId('672b4a8854ec8225b09293b2'),
ObjectId('672b4a8854ec8225b09293b3'), ObjectId('672b4a8854ec8225b09293b4'), ObjectId('672b4a8854ec8225b09293b5'), ObjectId('672b4a8854ec8225b09293b6'), ObjectId('672b4a8854ec8225b09293b7'), ObjectId('672b4a8854ec8225b09293b8'),
ObjectId('672b4a8854ec8225b09293b9'), ObjectId('672b4a8854ec8225b09293ba'), ObjectId('672b4a8854ec8225b09293bb'), ObjectId('672b4a8854ec8225b09293bc'), ObjectId('672b4a8854ec8225b09293bd'), ObjectId('672b4a8854ec8225b09293be'),
ObjectId('672b4a8854ec8225b09293bf'), ObjectId('672b4a8854ec8225b09293c0'), ObjectId('672b4a8854ec8225b09293c1'), ObjectId('672b4a8854ec8225b09293c2'), ObjectId('672b4a8854ec8225b09293c3'), ObjectId('672b4a8854ec8225b09293c4'),
ObjectId('672b4a8854ec8225b09293c5'), ObjectId('672b4a8854ec8225b09293c6'), ObjectId('672b4a8854ec8225b09293c7'), ObjectId('672b4a8854ec8225b09293c8'), ObjectId('672b4a8854ec8225b09293c9'), ObjectId('672b4a8854ec8225b09293ca'),
ObjectId('672b4a8854ec8225b09293cb'), ObjectId('672b4a8854ec8225b09293cc'), ObjectId('672b4a8854ec8225b09293cd'), ObjectId('672b4a8854ec8225b09293ce'), ObjectId('672b4a8854ec8225b09293cf'), ObjectId('672b4a8854ec8225b09293d0'),
ObjectId('672b4a8854ec8225b09293d1'), ObjectId('672b4a8854ec8225b09293d2'), ObjectId('672b4a8854ec8225b09293d3'), ObjectId('672b4a8854ec8225b09293d4'), ObjectId('672b4a8854ec8225b09293d5'), ObjectId('672b4a8854ec8225b09293d6'),
```


```
print("Data successfully stored in MongoDB!")
```

```
Data successfully stored in MongoDB!
```



Machine Learning Model

A machine learning model (possibly a regression model, given the focus on salary prediction) is implemented to predict the ConvertedCompYearly variable. The process includes:

1. **Model Selection**: Selection of an appropriate algorithm, potentially involving a comparison of several algorithms to determine the best fit.
2. **Training**: Training the model on the prepared dataset.

3. **Evaluation**: Evaluating model performance using standard metrics such as Mean Absolute Error (MAE) or Mean Squared Error (MSE) to gauge accuracy in salary prediction.
4. **Hyper parameter Tuning**: Fine-tuning model parameters to improve performance.

```
from sklearn.tree import DecisionTreeRegressor
dec_tree_reg = DecisionTreeRegressor(random_state=0)
dec_tree_reg.fit(X, y.values)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

```
y_pred = dec_tree_reg.predict(X)
```

```
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

```
$31,646.22
```

```
from sklearn.ensemble import RandomForestRegressor
random_forest_reg = RandomForestRegressor(random_state=0)
random_forest_reg.fit(X, y.values)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

```
y_pred = random_forest_reg.predict(X)
```

```
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

```
$31,776.30
```

```
from sklearn.model_selection import GridSearchCV

max_depth = [None, 2,4,6,8,10,12]
parameters = {"max_depth": max_depth}

regressor = DecisionTreeRegressor(random_state=0)
gs = GridSearchCV(regressor, parameters, scoring='neg_mean_squared_error')
gs.fit(X, y.values)
```

```
GridSearchCV
best_estimator_: DecisionTreeRegressor
DecisionTreeRegressor
```

```
regressor = gs.best_estimator_

regressor.fit(X, y.values)
y_pred = regressor.predict(X)
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

```
$33,603.23
```

Models Tested

1. **Decision Tree Regressor**:
   - This model partitions the data recursively based on feature values, creating a tree structure. It's known for its interpretability and ability to capture non-linear relationships.
   - **RMSE**: $31,646.22

2. **Random Forest Regressor**:
    - An ensemble model that creates multiple decision trees and averages their predictions, which generally improves accuracy and reduces overfitting.
    - **RMSE**: $31,776.30
3. **Decision Tree Regressor (Hyperparameter Tuning)**:
    - Using **GridSearchCV**, we tuned the `max_depth` parameter of the Decision Tree Regressor to find the best-performing tree depth.
    - **RMSE**: $33,683.23

Best Model Selection

Based on the RMSE values, the **Decision Tree Regressor** without hyperparameter tuning had the lowest error, with an RMSE of $31,646.22, making it the most accurate model for our dataset. Although the Random Forest Regressor usually performs well, in this case, the simpler Decision Tree Regressor performed slightly better.

Predictions

We used the **Decision Tree Regressor** model to make predictions on the test dataset. Below are the prections

Example-1

```
# country, edlevel, yearscode
X = np.array([["United States", 'Master’s degree', 15 ]])
X
```

```
array([['United States', 'Master’s degree', '15']], dtype='<U21')
```

```
# Fit the encoders on all unique values in the columns
le_country.fit(X[:, 0])
le_education.fit(X[:, 1])

# Transform the columns
X[:, 0] = le_country.transform(X[:, 0])
X[:, 1] = le_education.transform(X[:, 1])

# Convert to float if needed
X = X.astype(float)
X
```

```
array([[ 0.,  0., 15.]])
```

```
y_pred = regressor.predict(X)
y_pred
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarn
  warnings.warn(
array([101129.91304348])
```

Example-2

```
# country, edlevel, yearscode
X = np.array([["United States", 'Master’s degree', 5 ]])
X
```

```
array([['United States', 'Master’s degree', '5']], dtype='<U21')
```

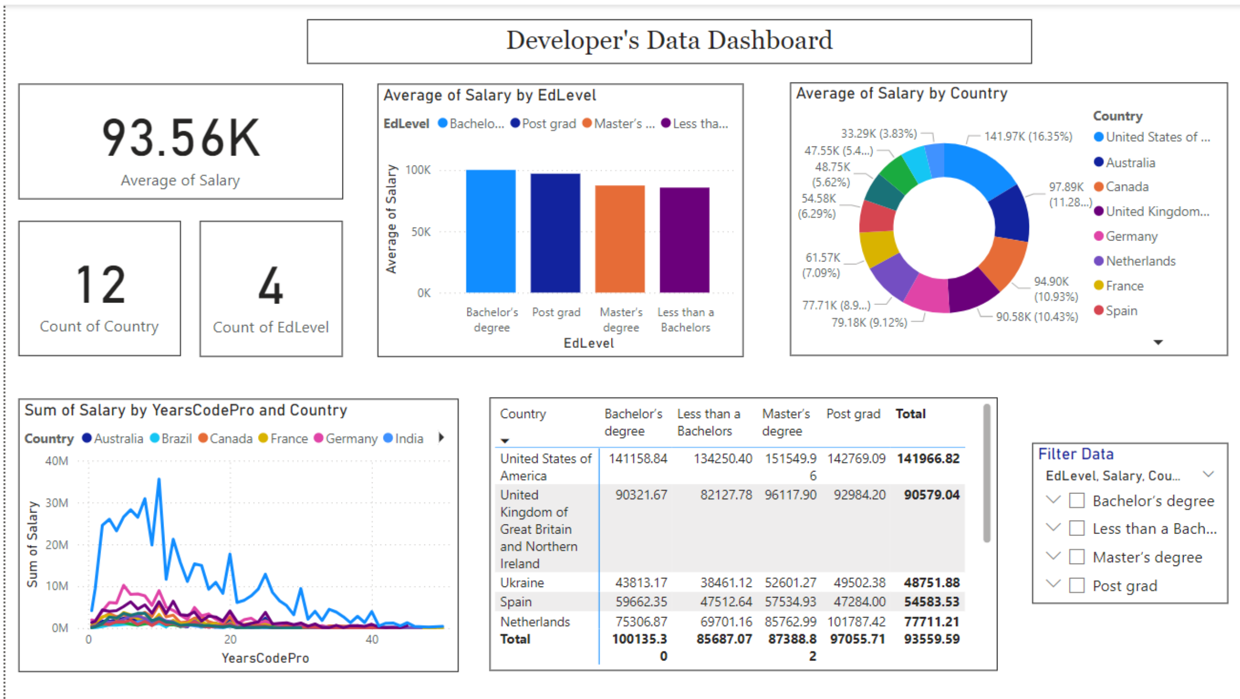
```
# Fit the encoders on all unique values in the columns
le_country.fit(X[:, 0])
le_education.fit(X[:, 1])

# Transform the columns
X[:, 0] = le_country.transform(X[:, 0])
X[:, 1] = le_education.transform(X[:, 1])

# Convert to float if needed
X = X.astype(float)
X
```

```
array([[0., 0., 5.]])
```

```
y_pred = regressor.predict(X)
y_pred
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWar
  warnings.warn(
array([84275.5])
```

With the help of PowerBI, Dashboard has been created to analyze insights.



Country	Bachelor's degree	Less than a Bachelors	Master's degree	Post grad	Total
United States of America	141158.84	134250.40	151549.96	142769.09	**141966.82**
United Kingdom of Great Britain and Northern Ireland	90321.67	82127.78	96117.90	92984.20	**90579.04**
Ukraine	43813.17	38461.12	52601.27	49502.38	**48751.88**
Spain	59662.35	47512.64	57534.93	47284.00	**54583.53**
Netherlands	75306.87	69701.16	85762.99	101787.42	**77711.21**
Total	**100135.30**	**85687.07**	**87388.82**	**97055.71**	**93559.59**

DATA FROM WEB SCRAPING

Introduction

The purpose of this project is to predict car prices based on data scraped from the PakWheels website. PakWheels is a popular platform for buying and selling vehicles, and it provides valuable insights into car market trends in Pakistan. This project aims to gather data from PakWheels, process it, store it in a database, and then use machine learning models to predict car prices based on features such as model, year, and other car specifications.

Data Extraction

To collect data from PakWheels, we used web scraping techniques to gather information from car listings. This process involved:

- **HTML Tags**: We navigated through the HTML structure of the PakWheels website to locate relevant data elements. Each car listing had specific tags for information like model, year, price, and contact details. By examining these tags, we identified where each piece of data was stored within the HTML document.
- **`strip()` Function**: The `strip()` function was used to clean up the text data extracted from these tags, removing unnecessary whitespace and formatting issues. This helped in making the data consistent and ready for further processing.

This extraction process resulted in a dataset with essential car details, either showing a price directly or indicating that the user should call for price information.

Data Transformation

After extracting the raw data from PakWheels, several data transformation steps were performed to clean and prepare the dataset for analysis and modeling. This process included **handling missing values**, **splitting the data into subsets**, and **applying label encoding** to categorical variables.

Handling Missing Values

The dataset contained some missing values, especially in fields like price (when listings prompted users to call for the price) and certain car specifications. To manage this:

1. **Price Column**: Listings without explicit price values were categorized separately into the `data_call` subset (listings with “call" instead of a price). Listings with price values were placed into the `data_update` subset, which was used for predictive modeling.
2. **Other Missing Values**: For missing values in other columns, such as car specifications, we either imputed these values with a suitable placeholder (such as "Unknown") or dropped rows with extensive missing data, depending on the importance of the feature for our analysis.

This process ensured that the data was clean and ready for machine learning, with minimal missing information that could interfere with model training.

Label Encoding

To enable machine learning algorithms to process categorical data, we applied **label encoding** to relevant categorical features. Label encoding is a technique that converts categorical variables (text) into numerical values, allowing models to interpret them effectively.

1. **Car Model and Brand**: Car models and brands were label-encoded so that each unique model and brand was represented by a unique numerical identifier.
2. **Fuel Type, Transmission, and Condition**: Other categorical fields, such as fuel type (e.g., Petrol, Diesel), transmission (e.g., Manual, Automatic), and car condition (e.g., New, Used), were also label-encoded.

Label encoding helped to transform the data into a numerical format, making it compatible with the machine learning algorithms while preserving the distinct categories.

```
[20] data_update.head()
```

	Manufacturer	Varient	Details	Location	Model	Distance Travelled	Fuel Type	Engine Capacity	Transmission	Featured	Price in PKR
0	29	103	819	175	2020	72000	5	1800.0	1	1	5800000.0
1	64	295	1104	140	2010	139000	5	1000.0	1	1	2100000.0
2	29	103	819	109	2019	52000	5	1800.0	1	1	6050000.0
3	29	102	343	140	2019	64000	5	1500.0	1	1	4125000.0
4	64	327	718	66	2022	18500	3	1200.0	1	1	9000000.0

Data Splitting

Following data cleaning and encoding, we split the dataset into two main subsets:

1. **`data_call`**: This subset contained listings where users were prompted to "call" for price information. Since it did not contain actual price values, this subset was not used for model training but could be retained for future analysis.
2. **`data_update`**: This subset contained listings with explicit price values. It served as the primary dataset for machine learning, as it provided the target variable (`price`) necessary for model training.

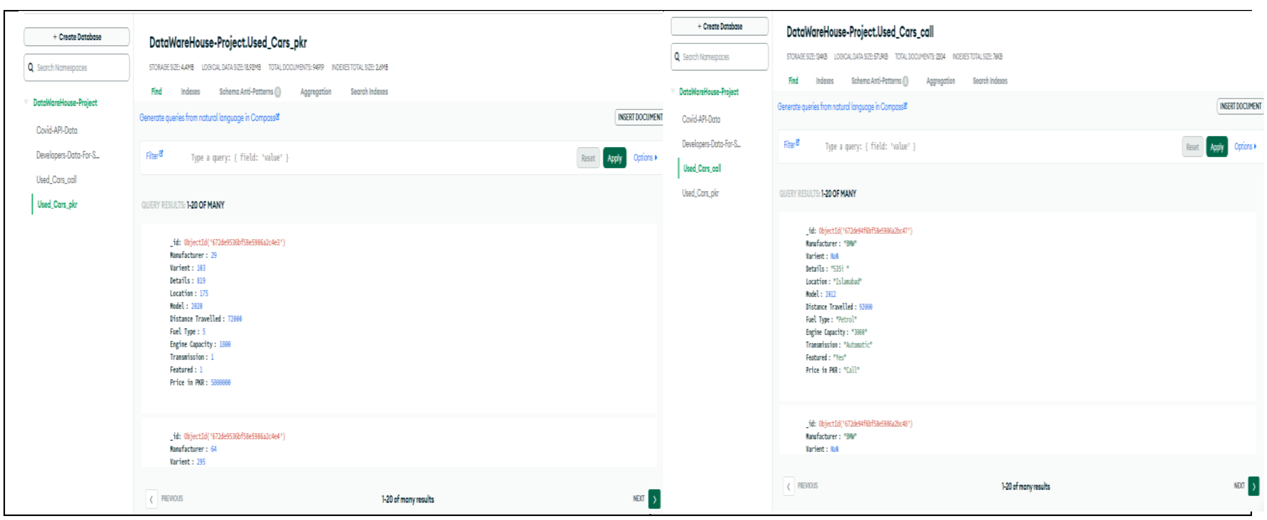
Data Load

The transformed data was stored in MongoDB for easy access and scalability. We created two separate **collections** in MongoDB to store each subset:

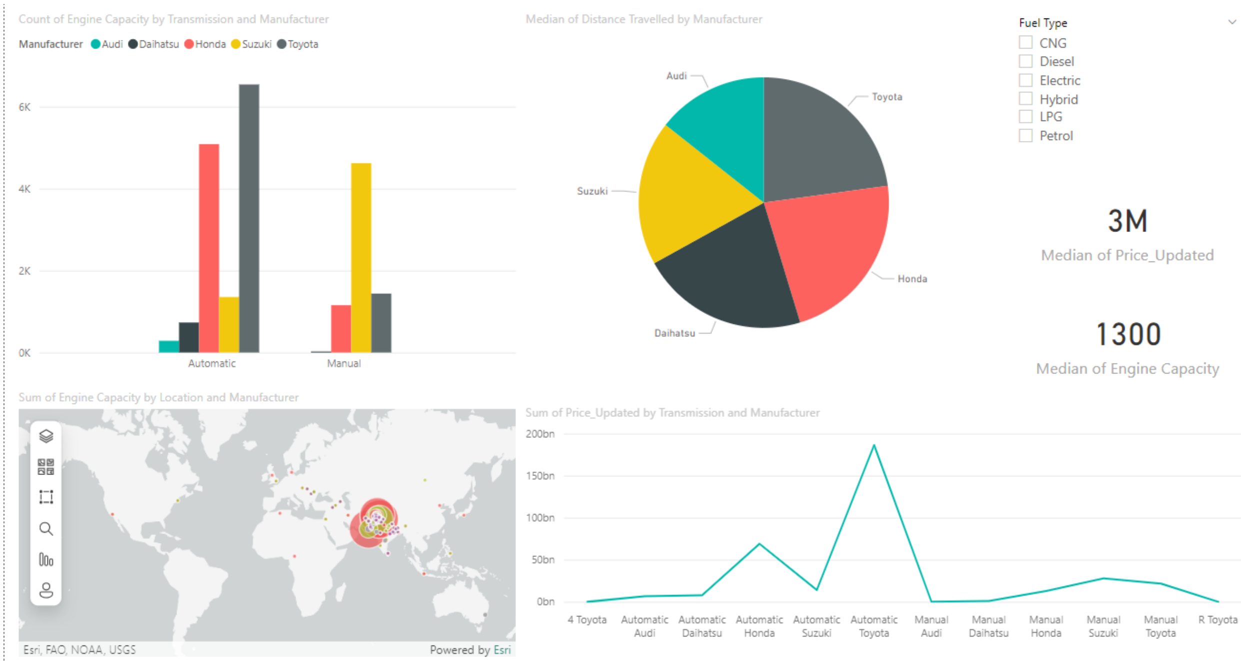
```python
# Insert data_call into the 'Used_Cars_call' collection
db['Used_Cars_call'].insert_many(data_call.to_dict("records"))

# Insert data_update into the 'Used_Cars_pkr' collection
db['Used_Cars_pkr'].insert_many(data_update.to_dict("records"))
```


ObjectId('672de9546bf58e5986a43552'), ObjectId('672de9546bf58e5986a43553'), ObjectId('672de9546bf58e5986a43554'), ObjectId('672de9546bf58e5986a43555'), ObjectId('672de9546bf58e5986a43556'),
ObjectId('672de9546bf58e5986a43557'), ObjectId('672de9546bf58e5986a43558'), ObjectId('672de9546bf58e5986a43559'), ObjectId('672de9546bf58e5986a4355a'), ObjectId('672de9546bf58e5986a4355b'),
ObjectId('672de9546bf58e5986a4355c'), ObjectId('672de9546bf58e5986a4355d'), ObjectId('672de9546bf58e5986a4355e'), ObjectId('672de9546bf58e5986a4355f'), ObjectId('672de9546bf58e5986a43560'),
ObjectId('672de9546bf58e5986a43561'), ObjectId('672de9546bf58e5986a43562'), ObjectId('672de9546bf58e5986a43563'), ObjectId('672de9546bf58e5986a43564'), ObjectId('672de9546bf58e5986a43565'),
ObjectId('672de9546bf58e5986a43566'), ObjectId('672de9546bf58e5986a43567'), ObjectId('672de9546bf58e5986a43568'), ObjectId('672de9546bf58e5986a43569'), ObjectId('672de9546bf58e5986a4356a'),
ObjectId('672de9546bf58e5986a4356b'), ObjectId('672de9546bf58e5986a4356c'), ObjectId('672de9546bf58e5986a4356d'), ObjectId('672de9546bf58e5986a4356e'), ObjectId('672de9546bf58e5986a4356f'),
ObjectId('672de9546bf58e5986a43570'), ObjectId('672de9546bf58e5986a43571'), ObjectId('672de9546bf58e5986a43572'), ObjectId('672de9546bf58e5986a43573'), ObjectId('672de9546bf58e5986a43574'),
ObjectId('672de9546bf58e5986a43575'), ObjectId('672de9546bf58e5986a43576'), ObjectId('672de9546bf58e5986a43577'), ObjectId('672de9546bf58e5986a43578'), ObjectId('672de9546bf58e5986a43579'),
ObjectId('672de9546bf58e5986a4357a'), ObjectId('672de9546bf58e5986a4357b'), ObjectId('672de9546bf58e5986a4357c'), ObjectId('672de9546bf58e5986a4357d'), ObjectId('672de9546bf58e5986a4357e'),
ObjectId('672de9546bf58e5986a4357f'), ObjectId('672de9546bf58e5986a43580'), ObjectId('672de9546bf58e5986a43581'), ObjectId('672de9546bf58e5986a43582'), ObjectId('672de9546bf58e5986a43583'),
ObjectId('672de9546bf58e5986a43584'), ObjectId('672de9546bf58e5986a43585'), ObjectId('672de9546bf58e5986a43586'), ObjectId('672de9546bf58e5986a43587'), ObjectId('672de9546bf58e5986a43588'),
ObjectId('672de9546bf58e5986a43589'), ObjectId('672de9546bf58e5986a4358a'), ObjectId('672de9546bf58e5986a4358b'), ObjectId('672de9546bf58e5986a4358c'), ObjectId('672de9546bf58e5986a4358d'),
ObjectId('672de9546bf58e5986a4358e'), ObjectId('672de9546bf58e5986a4358f'), ObjectId('672de9546bf58e5986a43590'), ObjectId('672de9546bf58e5986a43591'), ObjectId('672de9546bf58e5986a43592'),
ObjectId('672de9546bf58e5986a43593'), ObjectId('672de9546bf58e5986a43594'), ObjectId('672de9546bf58e5986a43595'), ObjectId('672de9546bf58e5986a43596'), ObjectId('672de9546bf58e5986a43597'),
ObjectId('672de9546bf58e5986a43598'), ObjectId('672de9546bf58e5986a43599'), ObjectId('672de9546bf58e5986a4359a'), ObjectId('672de9546bf58e5986a4359b'), ObjectId('672de9546bf58e5986a4359c'),
ObjectId('672de9546bf58e5986a4359d'), ObjectId('672de9546bf58e5986a4359e'), ObjectId('672de9546bf58e5986a4359f'), ObjectId('672de9546bf58e5986a435a0'), ObjectId('672de9546bf58e5986a435a1'),
ObjectId('672de9546bf58e5986a435a2'), ObjectId('672de9546bf58e5986a435a3'), ObjectId('672de9546bf58e5986a435a4'), ObjectId('672de9546bf58e5986a435a5'), ObjectId('672de9546bf58e5986a435a6'),
ObjectId('672de9546bf58e5986a435a7'), ObjectId('672de9546bf58e5986a435a8'), ObjectId('672de9546bf58e5986a435a9'), ObjectId('672de9546bf58e5986a435aa'), ObjectId('672de9546bf58e5986a435ab'),
ObjectId('672de9546bf58e5986a435ac'), ObjectId('672de9546bf58e5986a435ad'), ObjectId('672de9546bf58e5986a435ae'), ObjectId('672de9546bf58e5986a435af'), ObjectId('672de9546bf58e5986a435b0'),
ObjectId('672de9546bf58e5986a435b1'), ObjectId('672de9546bf58e5986a435b2'), ObjectId('672de9546bf58e5986a435b3'), ObjectId('672de9546bf58e5986a435b4'), ObjectId('672de9546bf58e5986a435b5'),
ObjectId('672de9546bf58e5986a435b6'), ObjectId('672de9546bf58e5986a435b7'), ObjectId('672de9546bf58e5986a435b8'), ObjectId('672de9546bf58e5986a435b9'), ObjectId('672de9546bf58e5986a435ba'),
ObjectId('672de9546bf58e5986a435bb'), ObjectId('672de9546bf58e5986a435bc'), ObjectId('672de9546bf58e5986a435bd'), ObjectId('672de9546bf58e5986a435be'), ObjectId('672de9546bf58e5986a435bf'),
ObjectId('672de9546bf58e5986a435c0'), ObjectId('672de9546bf58e5986a435c1'), ObjectId('672de9546bf58e5986a435c2'), ObjectId('672de9546bf58e5986a435c3'), ObjectId('672de9546bf58e5986a435c4'),
ObjectId('672de9546bf58e5986a435c5'), ObjectId('672de9546bf58e5986a435c6'), ObjectId('672de9546bf58e5986a435c7'), ObjectId('672de9546bf58e5986a435c8'), ObjectId('672de9546bf58e5986a435c9'),
ObjectId('672de9546bf58e5986a435ca'), ObjectId('672de9546bf58e5986a435cb'), ObjectId('672de9546bf58e5986a435cc'), ObjectId('672de9546bf58e5986a435cd'), ObjectId('672de9546bf58e5986a435ce'),
ObjectId('672de9546bf58e5986a435cf'), ObjectId('672de9546bf58e5986a435d0'), ObjectId('672de9546bf58e5986a435d1'), ObjectId('672de9546bf58e5986a435d2'), ObjectId('672de9546bf58e5986a435d3'),
ObjectId('672de9546bf58e5986a435d4'), ObjectId('672de9546bf58e5986a435d5'), ObjectId('672de9546bf58e5986a435d6'), ObjectId('672de9546bf58e5986a435d7'), ObjectId('672de9546bf58e5986a435d8'),
ObjectId('672de9546bf58e5986a435d9'), ObjectId('672de9546bf58e5986a435da'), ObjectId('672de9546bf58e5986a435db'), ObjectId('672de9546bf58e5986a435dc'), ObjectId('672de9546bf58e5986a435dd'),
ObjectId('672de9546bf58e5986a435de'), ObjectId('672de9546bf58e5986a435df'), ObjectId('672de9546bf58e5986a435e0'), ObjectId('672de9546bf58e5986a435e1'), ObjectId('672de9546bf58e5986a435e2'),
ObjectId('672de9546bf58e5986a435e3'), ObjectId('672de9546bf58e5986a435e4'), ObjectId('672de9546bf58e5986a435e5'), ObjectId('672de9546bf58e5986a435e6'), ObjectId('672de9546bf58e5986a435e7'),
ObjectId('672de9546bf58e5986a435e8'), ObjectId('672de9546bf58e5986a435e9'), ObjectId('672de9546bf58e5986a435ea'), ObjectId('672de9546bf58e5986a435eb'), ObjectId('672de9546bf58e5986a435ec'),
ObjectId('672de9546bf58e5986a435ed'), ObjectId('672de9546bf58e5986a435ee'), ObjectId('672de9546bf58e5986a435ef'), ObjectId('672de9546bf58e5986a435f0'), ObjectId('672de9546bf58e5986a435f1'),
ObjectId('672de9546bf58e5986a435f2'), ObjectId('672de9546bf58e5986a435f3'), ObjectId('672de9546bf58e5986a435f4'), ObjectId('672de9546bf58e5986a435f5'), ObjectId('672de9546bf58e5986a435f6'),

- **`data_call` collection**: This collection contains the listings where users are prompted to call for price details.
- **`data_update` collection**: This collection stores listings with an explicit price, making it the primary dataset for our machine learning model.



Dashboard



Count of Engine Capacity by Transmission and Manufacturer
Manufacturer Audi Daihatsu Honda Suzuki Toyota
6K
4K
2K
0K
Automatic
Manual
Median of Distance Travelled by Manufacturer
Audi
Toyota
Suzuki
Honda
Daihatsu
Fuel Type
CNG
Diesel
Electric
Hybrid
LPG
Petrol
3M
Median of Price_Updated
1300
Median of Engine Capacity
Sum of Engine Capacity by Location and Manufacturer
Esri, FAO, NOAA, USGS
Powered by Esri
Sum of Price_Updated by Transmission and Manufacturer
200bn
150bn
100bn
50bn
0bn
4 Toyota
Automatic Audi
Automatic Daihatsu
Automatic Honda
Automatic Suzuki
Automatic Toyota
Manual Audi
Manual Daihatsu
Manual Honda
Manual Suzuki
Manual Toyota
R Toyota

DATA FROM API

Introduction

In today's data-driven landscape, centralized data repositories are crucial for deriving meaningful insights. This project involves creating a data warehouse using the ETL (Extract, Transform, Load) process to consolidate COVID-19 statistics from an external API. By extracting, transforming, and loading this data into MongoDB, we facilitate easy access and analysis of pandemic-related information.

Data Extraction

Data extraction is the first step in the ETL pipeline, where raw data is sourced from the COVID-19 API. This API provides comprehensive statistics on COVID-19, such as infection rates, recoveries, and testing figures by country.

Libraries Used: The requests library is used to make API calls, and json is utilized for parsing the API's JSON response, allowing smooth data retrieval.

```
covid = requests.get(url, headers=headers).json()

print(json.dumps(covid, indent=4))
```


```
{
    "get": "statistics",
    "parameters": [],
    "errors": [],
    "results": 238,
    "response": [
        {
            "continent": "Africa",
            "country": "Djibouti",
            "population": 1016097,
            "cases": {
                "new": null,
                "active": 74,
                "critical": null,
                "recovered": 15427,
                "1M_pop": "15441",
                "total": 15690
            },
            "deaths": {
                "new": null,
                "1M_pop": "186",
                "total": 189
            },
            "tests": {
                "1M_pop": "301094",
                "total": 305941
            },
            "day": "2024-11-06",
            "time": "2024-11-06T06:45:09+00:00"
        },
```


API URL: https://covid-193.p.rapidapi.com/statistics

Data Transformation

The transformation stage ensures that data is standardized, cleaned, and optimized for storage and querying. Key transformations include:

Data Cleaning: Using Pandas, missing values are handled, data types are corrected, and country names are standardized.

Schema Structuring: Data is organized into a structured format that aligns with MongoDB's document-based structure.

Libraries Used: Pandas for data manipulation and NumPy for numerical operations to ensure efficient data processing.

```
[ ] covid_data
```

	Country	Population	Recovered	TotalTests
0	China	1.448471e+09	379053.0	1.600000e+08
1	Falkland-Islands	3.539000e+03	1930.0	8.632000e+03
2	Montserrat	4.965000e+03	1376.0	1.776200e+04
3	Nauru	1.090300e+04	5347.0	2.050900e+04
4	Wallis-and-Futuna	1.098200e+04	438.0	2.050800e+04
...	...	...	...	...
165	USA	3.348053e+08	109814428.0	1.186852e+09
166	France	6.558452e+07	39970918.0	2.714902e+08
167	Germany	8.388360e+07	38240600.0	1.223324e+08
168	Brazil	2.153536e+08	36249161.0	6.377617e+07
169	S-Korea	5.132990e+07	34535939.0	1.580406e+07

170 rows × 4 columns

```
[ ] pakistan_data = covid_data[covid_data['Country'] == 'Pakistan']
```

```
[ ] pakistan_data
```

	Country	Population	Recovered	TotalTests
120	Pakistan	229488994.0	1538689.0	30589153.0

Data Load

In the final ETL phase, the transformed data is loaded into a MongoDB database. MongoDB's document-oriented model is well-suited for storing JSON-like data, making it ideal for handling the COVID-19 statistics.

Database: Data is stored in a MongoDB collection, providing a flexible and scalable storage solution.

Batch Loading: Data is inserted into MongoDB in batches, ensuring a smooth loading process and efficient access.

Libraries Used: pymongo for connecting to MongoDB and inserting data into collections

```
[ ] data_dict = covid_data.to_dict("records")  # Convert DataFrame to a list of dictionaries
    collection.insert_many(data_dict)
```


```
InsertManyResult([ObjectId('672127f31bbb924ee7847d4f'), ObjectId('672127f31bbb924ee7847d50'), ObjectId('672127f31bbb924ee7847d51'), ObjectId('672127f31bbb924ee7847d52'), ObjectId('672127f31bbb924ee7847d53'),
ObjectId('672127f31bbb924ee7847d54'), ObjectId('672127f31bbb924ee7847d55'), ObjectId('672127f31bbb924ee7847d56'), ObjectId('672127f31bbb924ee7847d57'), ObjectId('672127f31bbb924ee7847d58'), ObjectId('672127f31bbb924ee7847d59'),
ObjectId('672127f31bbb924ee7847d5a'), ObjectId('672127f31bbb924ee7847d5b'), ObjectId('672127f31bbb924ee7847d5c'), ObjectId('672127f31bbb924ee7847d5d'), ObjectId('672127f31bbb924ee7847d5e'), ObjectId('672127f31bbb924ee7847d5f'),
ObjectId('672127f31bbb924ee7847d60'), ObjectId('672127f31bbb924ee7847d61'), ObjectId('672127f31bbb924ee7847d62'), ObjectId('672127f31bbb924ee7847d63'), ObjectId('672127f31bbb924ee7847d64'), ObjectId('672127f31bbb924ee7847d65'),
ObjectId('672127f31bbb924ee7847d66'), ObjectId('672127f31bbb924ee7847d67'), ObjectId('672127f31bbb924ee7847d68'), ObjectId('672127f31bbb924ee7847d69'), ObjectId('672127f31bbb924ee7847d6a'), ObjectId('672127f31bbb924ee7847d6b'),
ObjectId('672127f31bbb924ee7847d6c'), ObjectId('672127f31bbb924ee7847d6d'), ObjectId('672127f31bbb924ee7847d6e'), ObjectId('672127f31bbb924ee7847d6f'), ObjectId('672127f31bbb924ee7847d70'), ObjectId('672127f31bbb924ee7847d71'),
ObjectId('672127f31bbb924ee7847d72'), ObjectId('672127f31bbb924ee7847d73'), ObjectId('672127f31bbb924ee7847d74'), ObjectId('672127f31bbb924ee7847d75'), ObjectId('672127f31bbb924ee7847d76'), ObjectId('672127f31bbb924ee7847d77'),
ObjectId('672127f31bbb924ee7847d78'), ObjectId('672127f31bbb924ee7847d79'), ObjectId('672127f31bbb924ee7847d7a'), ObjectId('672127f31bbb924ee7847d7b'), ObjectId('672127f31bbb924ee7847d7c'), ObjectId('672127f31bbb924ee7847d7d'),
ObjectId('672127f31bbb924ee7847d7e'), ObjectId('672127f31bbb924ee7847d7f'), ObjectId('672127f31bbb924ee7847d80'), ObjectId('672127f31bbb924ee7847d81'), ObjectId('672127f31bbb924ee7847d82'), ObjectId('672127f31bbb924ee7847d83'),
ObjectId('672127f31bbb924ee7847d84'), ObjectId('672127f31bbb924ee7847d85'), ObjectId('672127f31bbb924ee7847d86'), ObjectId('672127f31bbb924ee7847d87'), ObjectId('672127f31bbb924ee7847d88'), ObjectId('672127f31bbb924ee7847d89'),
ObjectId('672127f31bbb924ee7847d8a'), ObjectId('672127f31bbb924ee7847d8b'), ObjectId('672127f31bbb924ee7847d8c'), ObjectId('672127f31bbb924ee7847d8d'), ObjectId('672127f31bbb924ee7847d8e'), ObjectId('672127f31bbb924ee7847d8f'),
ObjectId('672127f31bbb924ee7847d90'), ObjectId('672127f31bbb924ee7847d91'), ObjectId('672127f31bbb924ee7847d92'), ObjectId('672127f31bbb924ee7847d93'), ObjectId('672127f31bbb924ee7847d94'), ObjectId('672127f31bbb924ee7847d95'),
ObjectId('672127f31bbb924ee7847d96'), ObjectId('672127f31bbb924ee7847d97'), ObjectId('672127f31bbb924ee7847d98'), ObjectId('672127f31bbb924ee7847d99'), ObjectId('672127f31bbb924ee7847d9a'), ObjectId('672127f31bbb924ee7847d9b'),
ObjectId('672127f31bbb924ee7847d9c'), ObjectId('672127f31bbb924ee7847d9d'), ObjectId('672127f31bbb924ee7847d9e'), ObjectId('672127f31bbb924ee7847d9f'), ObjectId('672127f31bbb924ee7847da0'), ObjectId('672127f31bbb924ee7847da1'),
ObjectId('672127f31bbb924ee7847da2'), ObjectId('672127f31bbb924ee7847da3'), ObjectId('672127f31bbb924ee7847da4'), ObjectId('672127f31bbb924ee7847da5'), ObjectId('672127f31bbb924ee7847da6'), ObjectId('672127f31bbb924ee7847da7'),
ObjectId('672127f31bbb924ee7847da8'), ObjectId('672127f31bbb924ee7847da9'), ObjectId('672127f31bbb924ee7847daa'), ObjectId('672127f31bbb924ee7847dab'), ObjectId('672127f31bbb924ee7847dac'), ObjectId('672127f31bbb924ee7847dad'),
ObjectId('672127f31bbb924ee7847dae'), ObjectId('672127f31bbb924ee7847daf'), ObjectId('672127f31bbb924ee7847db0'), ObjectId('672127f31bbb924ee7847db1'), ObjectId('672127f31bbb924ee7847db2'), ObjectId('672127f31bbb924ee7847db3'),
ObjectId('672127f31bbb924ee7847db4'), ObjectId('672127f31bbb924ee7847db5'), ObjectId('672127f31bbb924ee7847db6'), ObjectId('672127f31bbb924ee7847db7'), ObjectId('672127f31bbb924ee7847db8'), ObjectId('672127f31bbb924ee7847db9'),
ObjectId('672127f31bbb924ee7847dba'), ObjectId('672127f31bbb924ee7847dbb'), ObjectId('672127f31bbb924ee7847dbc'), ObjectId('672127f31bbb924ee7847dbd'), ObjectId('672127f31bbb924ee7847dbe'), ObjectId('672127f31bbb924ee7847dbf'),
ObjectId('672127f31bbb924ee7847dc0'), ObjectId('672127f31bbb924ee7847dc1'), ObjectId('672127f31bbb924ee7847dc2'), ObjectId('672127f31bbb924ee7847dc3'), ObjectId('672127f31bbb924ee7847dc4'), ObjectId('672127f31bbb924ee7847dc5'),
ObjectId('672127f31bbb924ee7847dc6'), ObjectId('672127f31bbb924ee7847dc7'), ObjectId('672127f31bbb924ee7847dc8'), ObjectId('672127f31bbb924ee7847dc9'), ObjectId('672127f31bbb924ee7847dca'), ObjectId('672127f31bbb924ee7847dcb'),
ObjectId('672127f31bbb924ee7847dcc'), ObjectId('672127f31bbb924ee7847dcd'), ObjectId('672127f31bbb924ee7847dce'), ObjectId('672127f31bbb924ee7847dcf'), ObjectId('672127f31bbb924ee7847dd0'), ObjectId('672127f31bbb924ee7847dd1'),
ObjectId('672127f31bbb924ee7847dd2'), ObjectId('672127f31bbb924ee7847dd3'), ObjectId('672127f31bbb924ee7847dd4'), ObjectId('672127f31bbb924ee7847dd5'), ObjectId('672127f31bbb924ee7847dd6'), ObjectId('672127f31bbb924ee7847dd7'),
ObjectId('672127f31bbb924ee7847dd8'), ObjectId('672127f31bbb924ee7847dd9'), ObjectId('672127f31bbb924ee7847dda'), ObjectId('672127f31bbb924ee7847ddb'), ObjectId('672127f31bbb924ee7847ddc'), ObjectId('672127f31bbb924ee7847ddd'),
ObjectId('672127f31bbb924ee7847dde'), ObjectId('672127f31bbb924ee7847ddf'), ObjectId('672127f31bbb924ee7847de0'), ObjectId('672127f31bbb924ee7847de1'), ObjectId('672127f31bbb924ee7847de2'), ObjectId('672127f31bbb924ee7847de3'),
ObjectId('672127f31bbb924ee7847de4'), ObjectId('672127f31bbb924ee7847de5'), ObjectId('672127f31bbb924ee7847de6'), ObjectId('672127f31bbb924ee7847de7'), ObjectId('672127f31bbb924ee7847de8'), ObjectId('672127f31bbb924ee7847de9'),
ObjectId('672127f31bbb924ee7847dea'), ObjectId('672127f31bbb924ee7847deb'), ObjectId('672127f31bbb924ee7847dec'), ObjectId('672127f31bbb924ee7847ded'), ObjectId('672127f31bbb924ee7847dee'), ObjectId('672127f31bbb924ee7847def'),
ObjectId('672127f31bbb924ee7847df0'), ObjectId('672127f31bbb924ee7847df1'), ObjectId('672127f31bbb924ee7847df2'), ObjectId('672127f31bbb924ee7847df3'), ObjectId('672127f31bbb924ee7847df4'), ObjectId('672127f31bbb924ee7847df5'),
ObjectId('672127f31bbb924ee7847df6'), ObjectId('672127f31bbb924ee7847df7'), ObjectId('672127f31bbb924ee7847df8')], acknowledged=True)
```