DataMidWare

A Data-Oriented Middleware & Application Integration

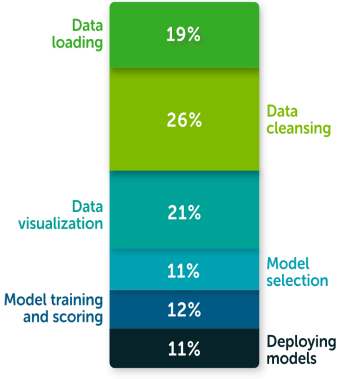
Jagriti Goswami

Outline

- Motivation
- Introduction of DataMidWare
- Architectural Overview of the Software
- Development Process Overview
- Code Architecture Representation using Component View
- Software, Tools, Libraries
- Code Implementation
- Testing
- Deployment & Execution
- Summary & Future Work

Motivation

- In this data-driven-age, every small and large organization is taking their business decisions based on the data insights.
- Transformation of raw data into usable cleaned data is crucial to take data-driven-decisions.
- But Data preparation is challenging for multiple reasons
  - Data from different sources come in various sizes and are different in nature.
  - Lack of integration between different data sources across the organization
  - Lack of proper integration between different technologies used for data science
  - Different platforms, software, libraries, and databases are used by developers
  - Managing dependencies and environments is another hurdle
  - Developing ad-hoc codes for specific use case
  - Bulk time is spent on data preparation (According to Anaconda survey-2020, on average **45%** of time is spent on data preparation, **21%** of time spent on visualization)



[The State of Data Science 2020]

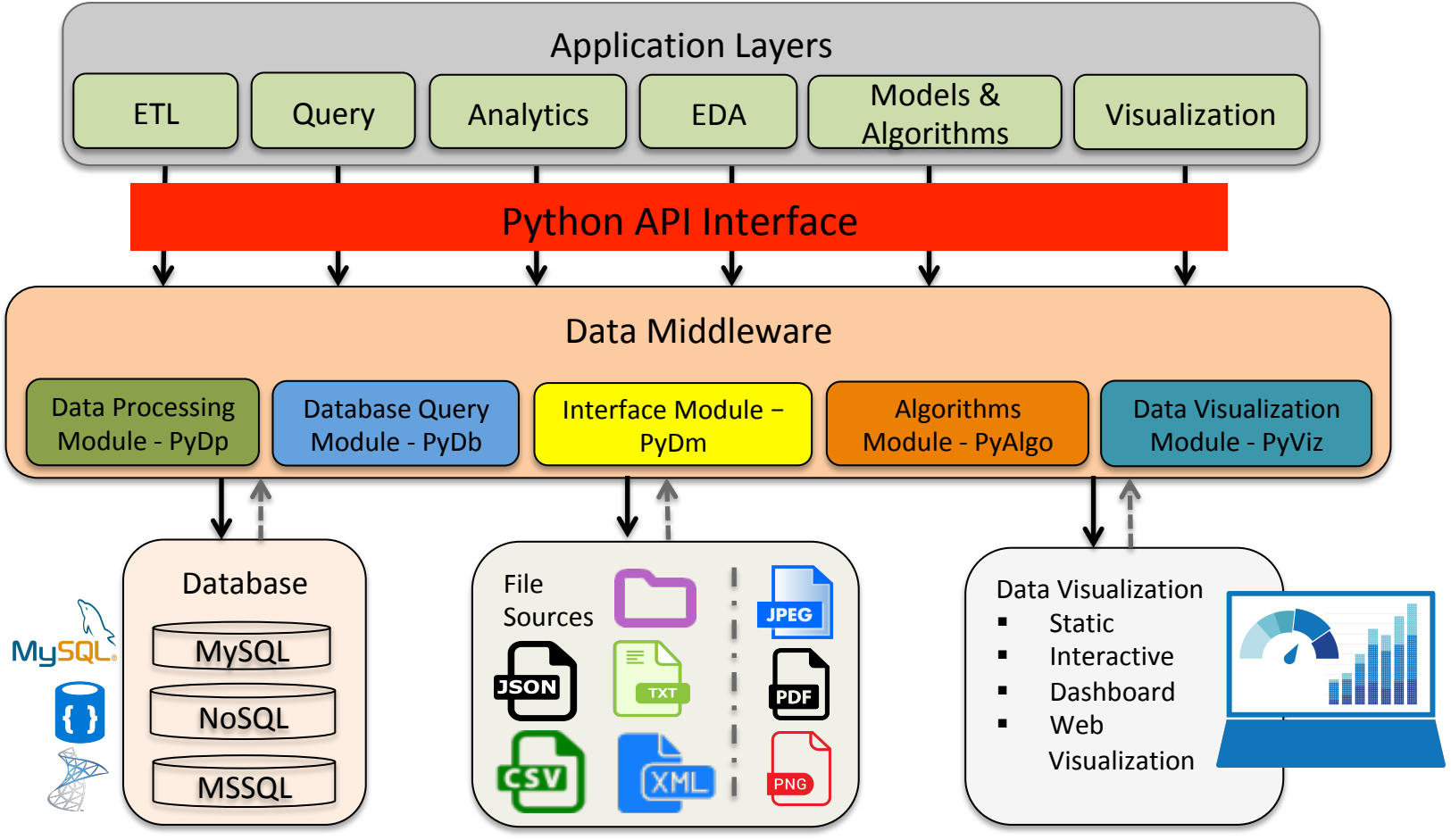
- Data cleaning and preparation takes valuable time away from real data science work − model building and deployment
- According to Anaconda's survey, **47%** of data scientists always use Python.

DataMidWare is a data middleware which accelerates data preparation, analysis, and visualization tasks by integrating different technologies, software, and libraries using its APIs implemented in Python 3.

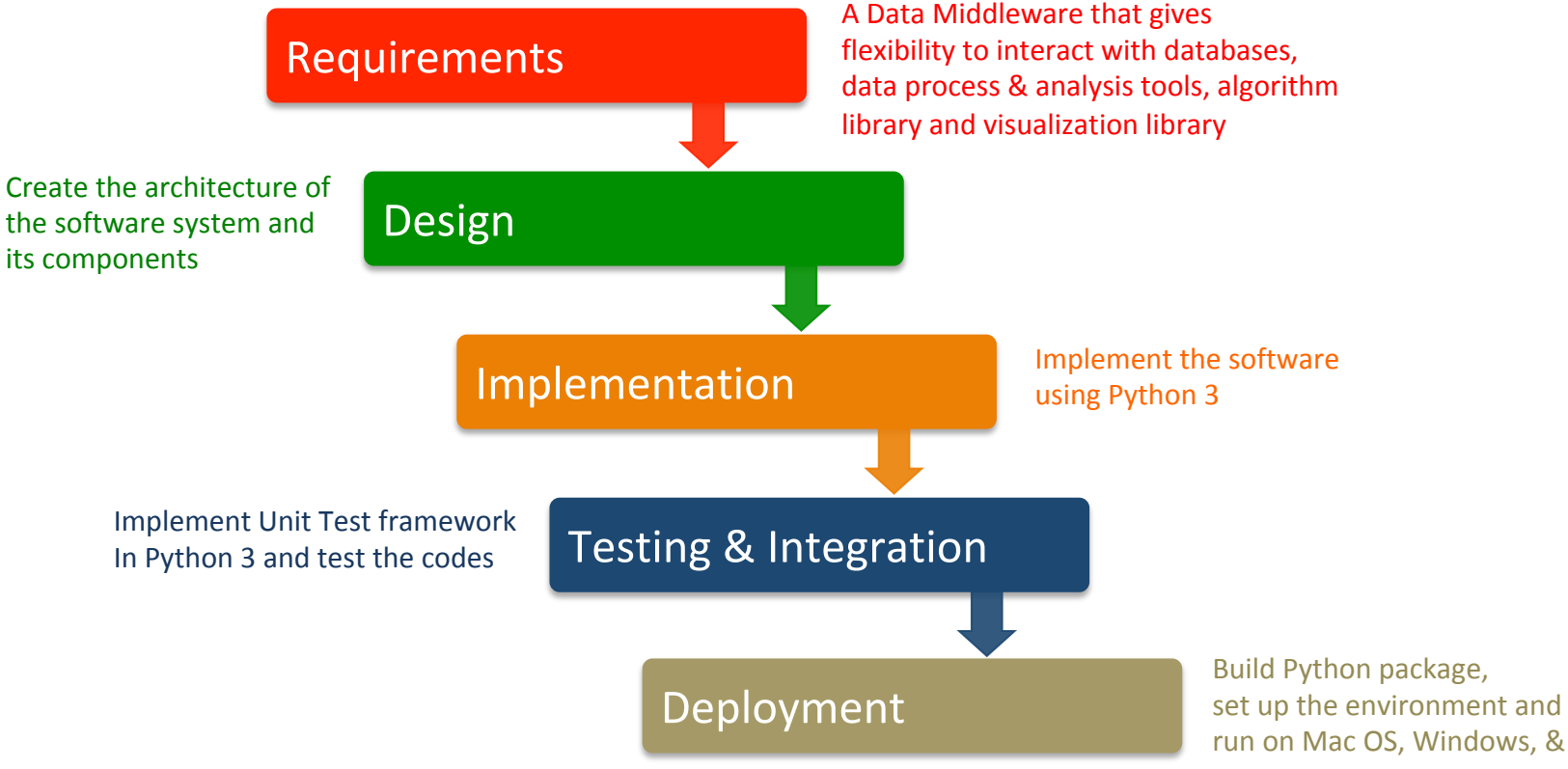
Introduction of DataMidWare

- **DataMidWare** - A data middleware implemented in Python 3.7.
- Provides direct interactions with database, data preparation & processing tool, algorithm library, and data visualization tool using its APIs.
- Imports and parses raw data (csv, json, txt, xml) from different sources and load to database (MySQL, NoSQL).
- Produces clean data from raw format and stores into database for analysis
- Performs SQL queries using its APIs
- Performs data analysis on the data stored in database
- Provides direct visualization of data stored in database
- Exports data in different format (csv, json) from database.
- *Accelerates data Preparation, Processing, and Visualization time*

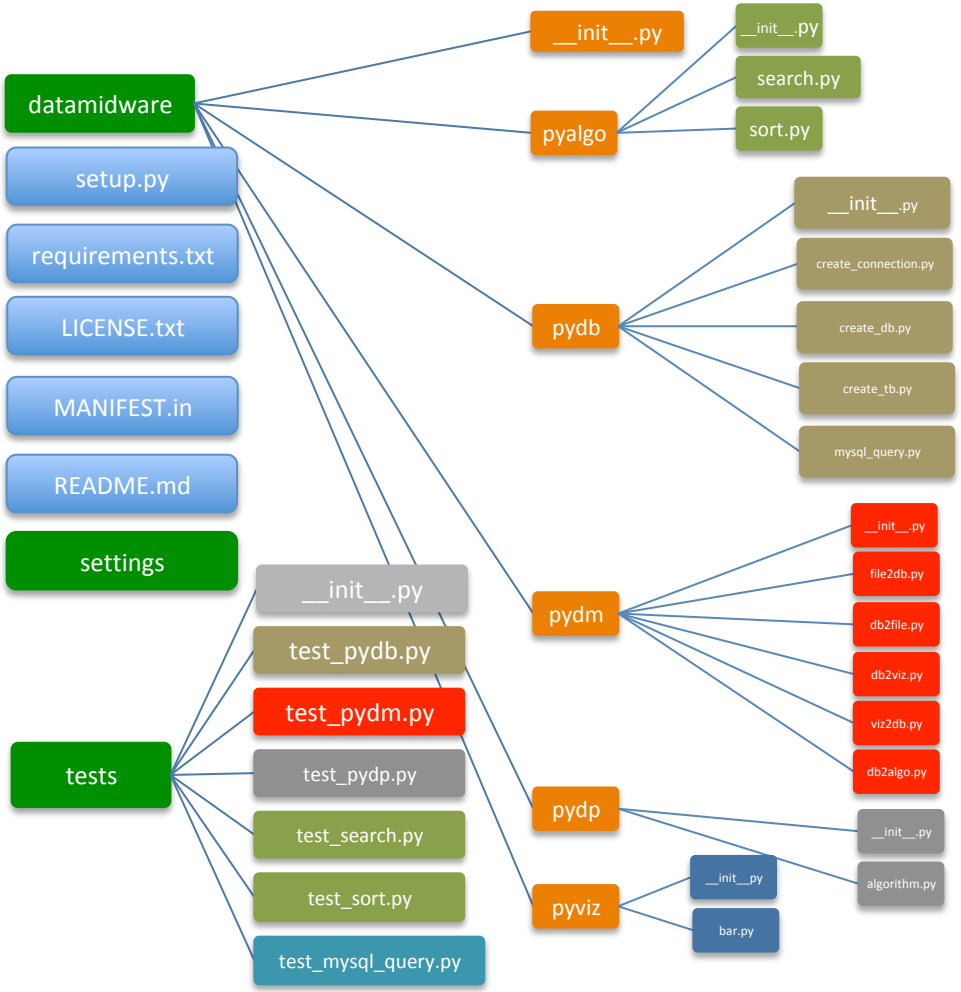
Architectural Overview



Development Process Overview



Code Architecture Representation using Component View



Software, Tools, Libraries

- IDE: PyCharm 18.3
- MySQL Workbench 6.3
- Programming language: Python (version 3.7)
- Python dependency packages

Package	Version
Numpy	1.19.2
Pandas	1.1.2
Plotly	4.11.0
PyMySQL	0.10.1
requests	2.24.0
SQLAlchemy	1.3.19
urllib3	1.25.10

requirements.txt

```
certifi==2020.6.20
chardet==3.0.4
idna==2.10
loguru==0.5.3
numpy==1.19.2
pandas==1.1.2
plotly==4.11.0
psutil==5.7.2
PyMySQL==0.10.1
python-dateutil==2.8.1
pytz==2020.1
requests==2.24.0
retrying==1.3.3
six==1.15.0
SQLAlchemy==1.3.19
urllib3==1.25.10
```

Code Implementation

API	Description	Parameters	Imports statement	Example Link
file2db.file2db(host, user, password, filename, db_name, tb_name, file_type="file_type", db_type="db_type")	Imports raw structured/ semi-structured data (csv, json) into database (MySQL, NoSQL)	**host**: host name **user**: user name **password**: password **filename**: filename to send to database **file_type**: file type (csv, json), str **db_type**: database type (mysql, nosql), str **tb_name**: table name where data will be stored	from datamidware.pydm import file2db	Example
db2file.db2file(host, user, password, file_path, db_name, tb_name, file_type="file_type", db_type="db_type")	Exports table data as csv/ json format from the database	**host**: host name **user**: user name **password**: password **file_path**: file path to export data from database **file_type**: file type (csv, json) **db_type**: database type (mysql, nosql) **db_name**: database from where data is exported **tb_name**: table from where data will be exported	from datamidware.pydm import db2file	Example
create_mysql_db.create_mysql_db(host, user, password, db_name)	Creates a new MySQL database	**host**: host name **user**: user name **password**: password **db_name**: database name to be created	from datamidware.pydm import create_mysql_db	
mysql_query.MySQLDatabase.select(tb_name, row_count="all")	Execute SQL query: SELECT * FROM table. Selecting all(or one if row_count="one") rows from the table.	**query**: SQL query to select rows: SELECT * FROM <table> **row_count**: "all" or "one" row. default "all". **return**: list of rows selected.	from datamidware.pydm import mysql_query	Example

Code Implementation

API	Description	Parameters	Imports statement	Example Link
csv2mysql.csv2mysql(host, user, password, filename, db_name, tb_name)	Imports csv file into mysql database	**host**: host name **user**: user name **password**: password **filename**: filename to send to database **db_name**: name of the database -- if database already exists, import data in the existing database, if not exists, create new database and import data. **tb_name**: name of the table - if table already exists, add data in the existing table, if not exists, create new table and import data.	from datamidware.pydm import csv2mysql	
mysql2csv.mysql2csv(host, user, password, file_path, db_name, tb_name)	Exports csv file from mysql database table	**host**: host name **user**: user name password: password **file_path**: file path to save csv file **db_name**: name of the database from where data will be exported **tb_name**: name of the table from where data will be exported	from datamidware.pydm import mysql2csv	
db2viz.db2viz(host, user, password, db_name, tb_name, kind=**None**, x=**None**, y=**None**, …)	Visualize the Database table data	**host**: host name, **user**: user name **password**: password, **db_name**: database name **tb_name**: table name, **kind**: plot kind (bar, horizontal bar, hist,..), **file_path**: file path to save figure **x**: x data; list or array-like, **y**: y data; list or array-like **title**: title of the figure, **label**: x-label, y-label	from datamidware.pydm import db2viz	Example

Testing

Modules	Description	Execution command	Remark
test_pydm.py	Test module for ***pydm*** sub-package	python3 -m unittest tests/test_pydm.py	**Pass**
test_mysql_query.py	Test module for ***mysql_query.py*** module	python3 -m unittest tests/test_mysql_query.py	**Pass**
test_search.py	Test module for ***search.py*** module	python3 -m unittest tests/test_search.py	**Pass**
test_sort.py	Test module for ***sort.py*** module	python3 -m unittest tests/test_sort.py	**Pass**

- To test all the modules, run all the above unittest command from the top-level directory
- To successfully run the test, config.ini file is required
- To write confitg.ini file go to /settings
- update write_config.py with database connection credentials, for e.g.,

(for MySQL database connection)

```
config_object["MYSQL"] = {
  "host": "hostname",
  "user": "username",
  "password": "password"
}
```

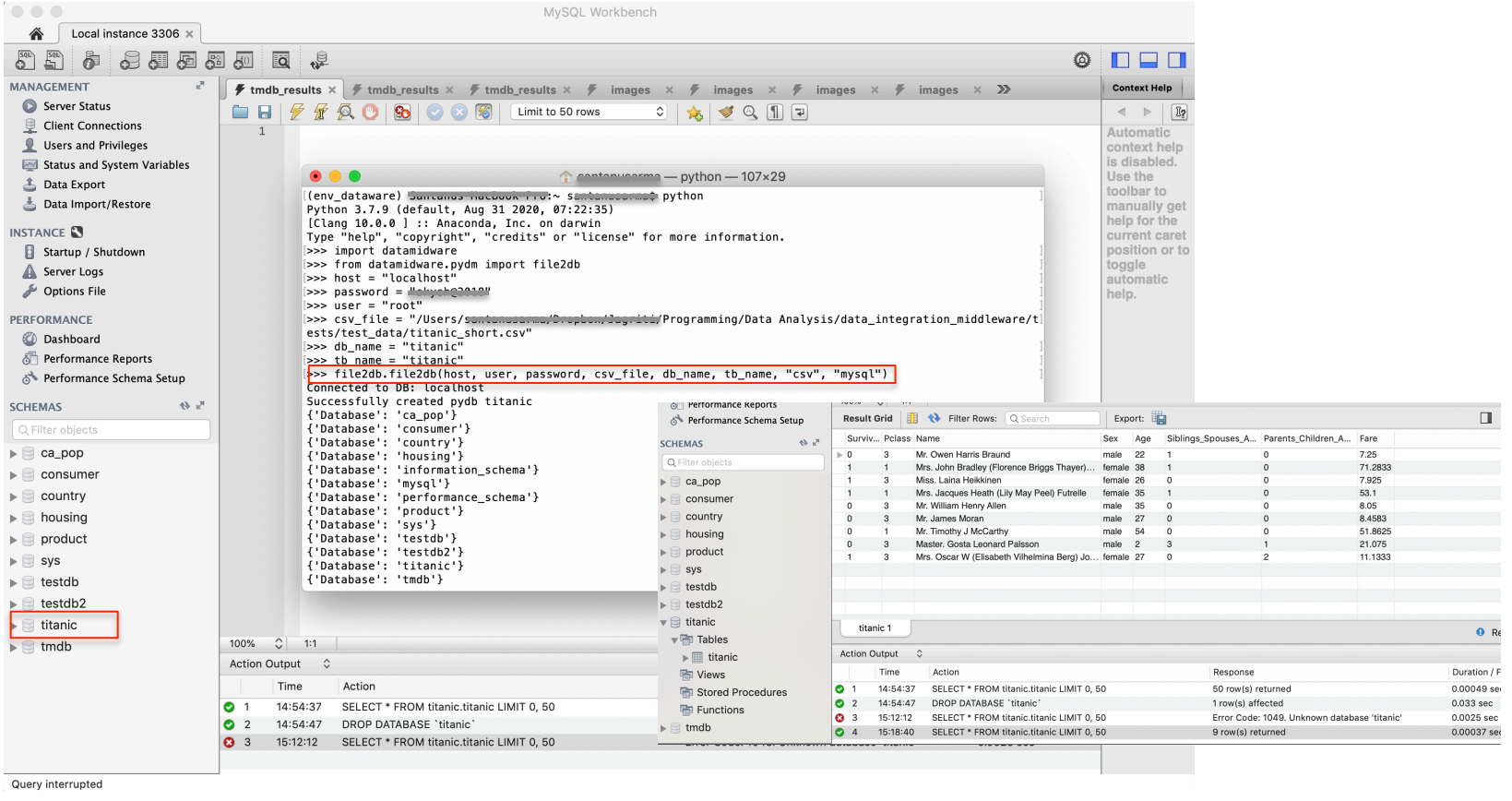
Deployment & Execution

- Using the terminal or an Anaconda Prompt, create a new environment – `conda create –n env_name python=3.7`
- Activate new environment using - `conda activate  env_name`
- Install package using pip – `pip install -i https://test.pypi.org/simple/ datamidware==0.0.11`
- Install requirements.txt using pip - `pip install –r requirements.txt`

```
(env_dataware) [redacted] pip install -i https://test.pypi.org/simple/
datamidware==0.0.11
Looking in indexes: https://test.pypi.org/simple/
Collecting datamidware==0.0.11
  Downloading https://test-files.pythonhosted.org/packages/34/1d/103524d61472838c103e1411f07c1263
ebf509e7a9f7242b2d72092a73be/datamidware-0.0.11-py3-none-any.whl (96 kB)
     |[redacted]| 96 kB 685 kB/s
Installing collected packages: datamidware
Successfully installed datamidware-0.0.11
(env_dataware) [redacted]$ pip install –r requirements.txt
ERROR: Invalid requirement: '–r'
(env_dataware) [redacted]$ pip install –r requirements.txt
Requirement already satisfied: certifi==2020.6.20 in ./anaconda3/envs/env_dataware/lib/python3.7/
site-packages (from –r requirements.txt (line 1)) (2020.6.20)
Requirement already satisfied: chardet==3.0.4 in ./anaconda3/envs/env_dataware/lib/python3.7/site
-packages (from –r requirements.txt (line 2)) (3.0.4)
Requirement already satisfied: idna==2.10 in ./anaconda3/envs/env_dataware/lib/python3.7/site-pac
kages (from –r requirements.txt (line 3)) (2.10)
Requirement already satisfied: loguru==0.5.3 in ./anaconda3/envs/env_dataware/lib/python3.7/site-
packages (from –r requirements.txt (line 4)) (0.5.3)
Requirement already satisfied: numpy==1.19.2 in ./anaconda3/envs/env_dataware/lib/python3.7/site-
packages (from –r requirements.txt (line 5)) (1.19.2)
Requirement already satisfied: pandas==1.1.2 in ./anaconda3/envs/env_dataware/lib/python3.7/site-
packages (from –r requirements.txt (line 6)) (1.1.2)
Requirement already satisfied: plotly==4.11.0 in ./anaconda3/envs/env_dataware/lib/python3.7/site
-packages (from –r requirements.txt (line 7)) (4.11.0)
Requirement already satisfied: psutil==5.7.2 in ./anaconda3/envs/env_dataware/lib/python3.7/site-
packages (from –r requirements.txt (line 8)) (5.7.2)
```

Example:***file2db.file2db***(host, user, password, filename, db_name, tb_name, file_type="file_type", db_type="db_type")

- Execute ***file2db.file2db***(host, user, password, filename, db_name, tb_name, file_type="file_type", db_type="db_type") to load csv file into MySQL DB
- Example data: titanic.csv



Back

Example:***db2file.db2file***(host, user, password, file_path, db_name, tb_name, file_type="file_type", db_type="db_type")

- Execute ***db2file.db2file***(host, user, password, file_path, db_name, tb_name, file_type="file_type", db_type="db_type") to export csv file from MySQL
- Example data: MySQL database name : titanic, table name = titanic

```
(env_dataware) Santanus-MacBook-Pro:~ santanusarma$ python
Python 3.7.9 (default, Aug 31 2020, 07:22:35)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import datamidware
>>> from datamidware.pydm import db2file
>>> host = "localhost"
>>> user = "root"
>>> password = "[redacted]"
>>> output_filepath = "/Users/santanusarma/Dropbox/Jagriti/Programming/Data Analysis/data_integration_middl
eware/tests/test_result/"
>>> db_name = "titanic"
>>> tb_name = "titanic"
>>> db2file.db2file(host, user, password, output_filepath, db_name, tb_name, "csv", "mysql")
2020-11-06 15:22:37,038 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIKE 'sql_mode'
2020-11-06 15:22:37,039 INFO sqlalchemy.engine.base.Engine {}
2020-11-06 15:22:37,042 INFO sqlalchemy.engine.bas
2020-11-06 15:22:37,042 INFO sqlalchemy.engine.bas
2020-11-06 15:22:37,044 INFO sqlalchemy.engine.bas
2020-11-06 15:22:37,044 INFO sqlalchemy.engine.bas
2020-11-06 15:22:37,045 INFO sqlalchemy.engine.bas
Collation` = 'utf8mb4_bin'
2020-11-06 15:22:37,045 INFO sqlalchemy.engine.bas
2020-11-06 15:22:37,049 INFO sqlalchemy.engine.bas
 anon_1
2020-11-06 15:22:37,049 INFO sqlalchemy.engine.bas
2020-11-06 15:22:37,049 INFO sqlalchemy.engine.bas
AS anon_1
2020-11-06 15:22:37,049 INFO sqlalchemy.engine.bas
2020-11-06 15:22:37,076 INFO sqlalchemy.engine.base.Engine SHOW CREATE TABLE `titanic`
2020-11-06 15:22:37,076 INFO sqlalchemy.engine.base.Engine {}
['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'Siblings_Spouses_Aboard', 'Parents_Children_Aboard', 'Fare']
2020-11-06 15:22:37,080 INFO sqlalchemy.engine.base.Engine DESCRIBE `SELECT * FROM titanic`
2020-11-06 15:22:37,080 INFO sqlalchemy.engine.base.Engine {}
2020-11-06 15:22:37,082 INFO sqlalchemy.engine.base.Engine ROLLBACK
2020-11-06 15:22:37,082 INFO sqlalchemy.engine.base.Engine SELECT * FROM titanic
2020-11-06 15:22:37,082 INFO sqlalchemy.engine.base.Engine {}
   Survived  Pclass  ... Parents_Children_Aboard     Fare
0         0       3  ...                       0   7.2500
1         1       1  ...                       0  71.2833
2         1       3  ...                       0   7.9250
3         1       1  ...                       0  53.1000
4         0       3  ...                       0   8.0500

[5 rows x 8 columns]
File, /Users/santanusarma/Dropbox/Jagriti/Programming/Data Analysis/data_integration_middleware/tests/test_
result/titanic.csv, has been created successfully
>>>
```

Back

Example: ***mysql_query.MySQLDatabase.select***(tb_name, row_count="all")

- Execute ***mysql_query.MySQLDatabase.select***(tb_name, row_count="all") to select all rows from MySQL table
- Example data: MySQL database name : titanic, table name = titanic

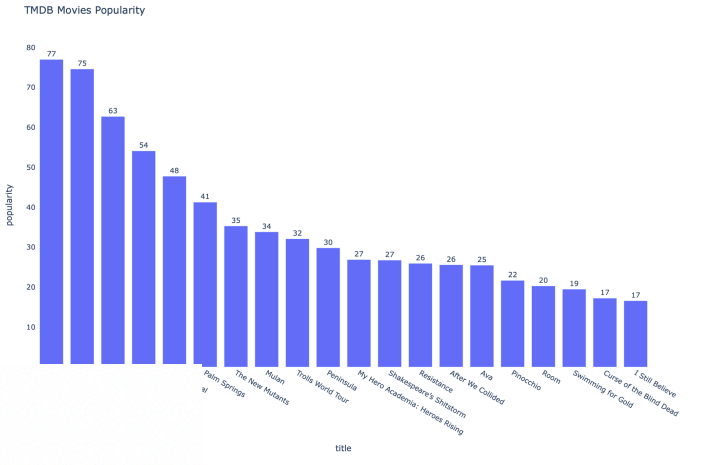
```
(env_dataware) [redacted] python
Python 3.7.9 (default, Aug 31 2020, 07:22:35)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import datamidware
>>> from datamidware.pydm import mysql_query
>>> host = "localhost"
>>> user = "root"
>>> password = "[redacted]"
>>> db_name = "titanic"
>>> tb_name = "titanic"
>>> db = mysql_query.MySQLDatabase(host, user, password, db_name)
>>> rows = db.select(tb_name, row_count="all")
2020-11-06 15:33:27.254 | INFO     | datamidware.pydm.mysql_query:open_connection:44 - Connection opened s
uccessfully.
2020-11-06 15:33:27.256 | INFO     | datamidware.pydm.mysql_query:select:78 - Database connection closed.
>>> rows
[{'Survived': 0, 'Pclass': 3, 'Name': 'Mr. Owen Harris Braund', 'Sex': 'male', 'Age': 22, 'Siblings_Spouse
s_Aboard': 1, 'Parents_Children_Aboard': 0, 'Fare': 7.25}, {'Survived': 1, 'Pclass': 1, 'Name': 'Mrs. John
 Bradley (Florence Briggs Thayer) Cumings', 'Sex': 'female', 'Age': 38, 'Siblings_Spouses_Aboard': 1, 'Par
ents_Children_Aboard': 0, 'Fare': 71.2833}, {'Survived': 1, 'Pclass': 3, 'Name': 'Miss. Laina Heikkinen',
'Sex': 'female', 'Age': 26, 'Siblings_Spouses_Aboard': 0, 'Parents_Children_Aboard': 0, 'Fare': 7.925}, {'
Survived': 1, 'Pclass': 1, 'Name': 'Mrs. Jacques Heath (Lily May Peel) Futrelle', 'Sex': 'female', 'Age':
35, 'Siblings_Spouses_Aboard': 1, 'Parents_Children_Aboard': 0, 'Fare': 53.1}, {'Survived': 0, 'Pclass': 3
, 'Name': 'Mr. William Henry Allen', 'Sex': 'male', 'Age': 35, 'Siblings_Spouses_Aboard': 0, 'Parents_Chil
dren_Aboard': 0, 'Fare': 8.05}, {'Survived': 0, 'Pclass': 3, 'Name': 'Mr. James Moran', 'Sex': 'male', 'Ag
e': 27, 'Siblings_Spouses_Aboard': 0, 'Parents_Children_Aboard': 0, 'Fare': 8.4583}, {'Survived': 0, 'Pcla
ss': 1, 'Name': 'Mr. Timothy J McCarthy', 'Sex': 'male', 'Age': 54, 'Siblings_Spouses_Aboard': 0, 'Parents
_Children_Aboard': 0, 'Fare': 51.8625}, {'Survived': 0, 'Pclass': 3, 'Name': 'Master. Gosta Leonard Palsso
n', 'Sex': 'male', 'Age': 2, 'Siblings_Spouses_Aboard': 3, 'Parents_Children_Aboard': 1, 'Fare': 21.075},
{'Survived': 1, 'Pclass': 3, 'Name': 'Mrs. Oscar W (Elisabeth Vilhelmina Berg) Johnson', 'Sex': 'female',
'Age': 27, 'Siblings_Spouses_Aboard': 0, 'Parents_Children_Aboard': 2, 'Fare': 11.1333}]
>>>
```

Back

Example: ***db2viz.db2viz***(host, user, password, db_name, tb_name, kind=**None**, x=**None**, y=**None**, …)

- Execute ***db2viz.db2viz***(host, user, password, db_name, tb_name, kind=**None**, x=**None**, y=**None**, …) to plot bar chart to visualize movies by popularity from tmdb_results table from MySQL database tmdb.
- Example data: database name - tmdb, table name - tmdb_results

```
>>> import datamidware
>>> from datamidware.pydm import db2viz
>>> host = "localhost"
>>> user = "root"
>>> password = "[redacted]"
>>> db_name = "tmdb"
>>> tb_name = "tmdb_results"
>>> db2viz.db2viz(host, user, password, db_name, tb_name, db_type="mysql", kind="bar", y="popularity", x="
title", title="TMDB Movies Popularity", labels={"y": "popularity", "x": "title"}, update_trace_text=True,
file_path="/User[redacted]/Programming/Data Analysis/data_integration_middleware/tests
/test_result/", save2db=dict(host="localhost", user="root", password="[redacted]", db_type="mysql", db_nam
e="tmdb", tb_name="image"), show=True)
2020-11-06 15:53:32,291 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIKE 'sql_mode'
2020-11-06 15:53:32,291 INFO sqlalchemy.engine.base.Engine {}
2020-11-06 15:53:32,293 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIKE 'lower_case_table_names'
2020-11-06 15:53:32,293 INFO sqlalchemy.engine.base.Engine {}
2020-11-06 15:53:32,294 INFO sqlalchemy.engine.base.Engine SELECT DATABASE()
```



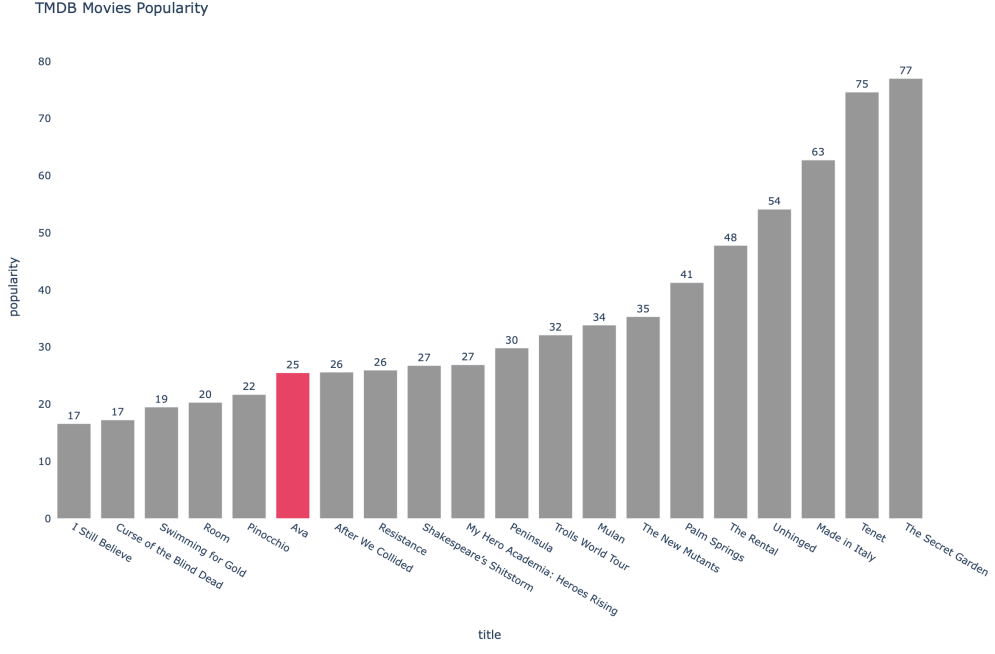

```
_result/tmdb_movies_popularity.png, has been created successfully
localhost root [redacted] tmdb image
('image',)
('tmdb_results',)
Inserting BLOB into table
Image inserted successfully as a BLOB into image table 1
MySQL connection is closed
```

Back

Example: ***db2viz.db2viz***(host, user, password, db_name, tb_name, kind=**None**, x=**None**, y=**None**, …)

- Execute ***db2viz.db2viz***(host, user, password, db_name, tb_name, kind=**None**, x=**None**, y=**None**, …) to plot bar chart for movies by popularity from tmdb_results table in ascending order and highlighting movie title - “Ava”

```
>>> db2viz.db2viz(host, user, password, db_name, tb_name, db_type="mysql", kind="bar", y="popularity", x="title", title="TMDB Movies Popularity", labels={"y": "popularity", "x": "title"}, update_trace_text=True, set_col_color="Ava", sort_asc=True, file_path="/[redacted]/Programming/Data Analysis/data_integration_middleware/tests/test_result/", save2db=dict(host="localhost", user="root", password="[redacted]", db_type="mysql", db_name="tmdb", tb_name="image"), show=True)
2020-11-06 16:02:53,064 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIKE 'sql_mode'
2020-11-06 16:02:53,064 INFO sqlalchemy.engine.base.Engine {}
2020-11-06 16:02:53,065 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIKE 'lower_case_table_names'
```

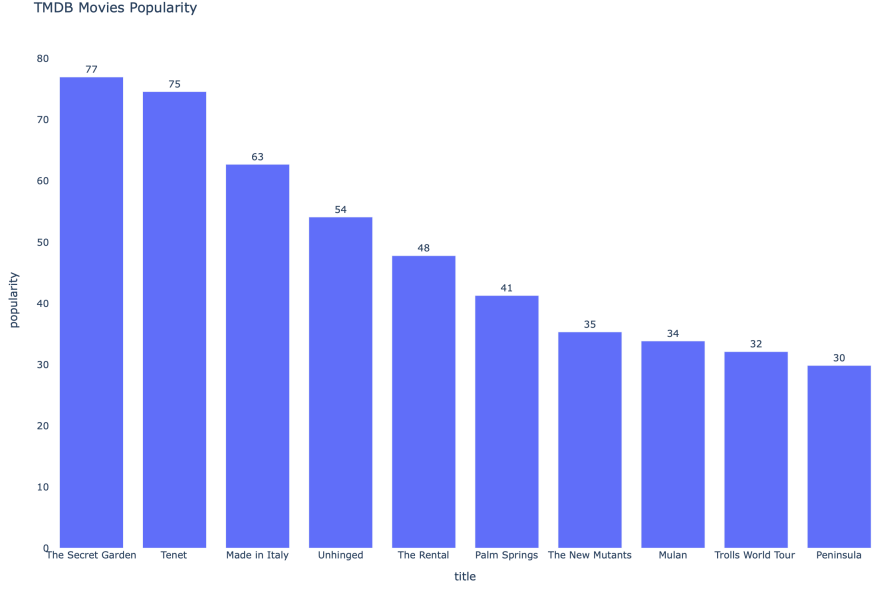


Back

Example: ***db2viz.db2viz***(host, user, password, db_name, tb_name, kind=**None**, x=**None**, y=**None**, …)

- Execute ***db2viz.db2viz***(host, user, password, db_name, tb_name, kind=**None**, x=**None**, y=**None**, …) to plot bar chart for 10 highest movies by popularity from tmdb_results table of tmdb database.

```
>>> db2viz.db2viz(host, user, password, db_name, tb_name, db_type="mysql", kind="bar", y="popularity", x="
title", title="TMDB Movies Popularity", labels={"y": "popularity", "x": "title"}, update_trace_text=True,
N_largest=10, file_path="/Users/[redacted]/Programming/Data Analysis/data_integration_mi
ddleware/tests/test_result/", save2db=dict(host="localhost", user="root", password="[redacted]", db_type="
mysql", db_name="tmdb", tb_name="image"), show=True)
2020-11-06 16:06:21,103 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIKE 'sql_mode'
2020-11-06 16:06:21,104 INFO sqlalchemy.engine.base.Engine {}
2020-11-06 16:06:21,105 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIKE 'lower_case_table_names'
2020-11-06 16:06:21,105 INFO sqlalchemy.engine.base.Engine {}
```



Back

Summary & Future Work

- Summary
  - A data middleware that gives flexibility to directly interact with database, data preparation & processing tool, algorithm library, and data visualization tool using its APIs; *accelerates data preparation, processing, & Visualization time*
  - Imports and parses raw data (csv, json, txt, xml) and load to database (MySQL, NoSQL).
  - Produces clean data from raw format and stores into database for analysis
  - Performs SQL queries
  - Provides data analysis and direct visualization of data stored in database
  - Exports data in different format (csv, json) from database.

- Future Work
  - Extend all the libraries – (data processing, algorithms, and data visualization library)
  - Integrate Exploratory and Inferential data analysis library
  - Integrate Machine Learning algorithms library
  - Integrate Web Visualization
  - Integrate with other Databases (MSSQL)
  - Integrate with Splunk