



Academic Year	Module	Assessment Number	Assessment Type
2024	Foundational Data Engineering	1	Report

## Summer Class Assignment

Student Id : 2408414

Student Name : Shashank Pandey

Section : L6CG20

Tutor : Gunjan Kumar Mishra

Submitted on : 09-06-2025.

## Table of Contents

Table of Contents .....	2
Week 1 – Data Extraction (Architecture Setup).....	3
Week 2 – Data Extraction (Continued) .....	7
Week 3- Created all the Schemas left (Staging, Target, Transform).....	11
Week 4- Data Loading.....	12
Week 5 – Orchestration with Prefect.....	17
Week 6 – Visualization (Metabase) .....	23
Conclusion .....	27

# Week 1 – Data Extraction (Architecture Setup)

## Step 1: AWS Setup

- Enrolled in AWS Academy and accessed the AWS console.
- Created an S3 bucket to store raw data (CSV & JSON).
- Configured public access for uploaded files (since IAM was restricted in AWS Academy).
- Uploaded raw datasets (CSV + JSON) into separate folders in the S3 bucket.

The screenshots show the AWS Academy Learner Lab interface. The top screenshot displays the initial setup screen with a sidebar containing links such as Environment Overview, Environment Navigation, Access the AWS Management Console, Region restriction, Service usage and other restrictions, Using the terminal in the browser, Running AWS CLI commands, Using the AWS SDK for Python, Preserving your budget, Accessing EC2 Instances, SSH Access to EC2 Instances, and SSH Access from Windows. The bottom screenshot shows the terminal window after starting the lab, displaying a command prompt (eee\_W\_4809154@runweb187238:~\$).

Console Home [Info](#)

Recently visited [Info](#)

S3

View all services

Applications (0) [Info](#)

Region: US East (N. Virginia)

Select Region: us-east-1 (Current Region) [Find applications](#)

Name Description Region Originat.

No applications  
Get started by creating an application.

Create application

Welcome to AWS [Info](#)

Getting started with AWS [Learn the fundamentals and](#)

AWS Health [Info](#)

Open issues 0 Past 7 days

Cost and usage [Info](#)

Current month Cost (\$) \$0.00 0

Reset to default layout + Add widgets

aws Search [Alt+S] Account ID: 8327-1226-3612 United States (N. Virginia) v vclabs/user4049511-S.Pandey25@wlv.ac.uk

Amazon S3 [Buckets](#) shawshank77

General purpose buckets

- Directory buckets
- Table buckets
- Vector buckets
- Access Grants
- Access Points (General Purpose Buckets, Fsx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

shawshank77 [Info](#)

Objects (2)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Find objects by prefix

Name	Type	Last modified	Size	Storage class
CSV/	Folder	-	-	-
JSON/	Folder	-	-	-

The figure consists of two vertically stacked screenshots of the AWS S3 console. Both screenshots show a bucket named 'shawshank77' containing three objects: 'customers.csv', 'products.csv', and 'sales.csv' (CSV files) and 'products.json', 'sales.json' (JSON files). The top screenshot is for 'CSV/' and the bottom one is for 'JSON/'. Each screenshot shows a left sidebar with navigation links like 'Amazon S3', 'General purpose buckets', and 'Storage Lens'. The main area displays a table of objects with columns for Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
customers.csv	csv	July 30, 2025, 10:51:21 (UTC+05:45)	4.8 KB	Standard
products.csv	csv	July 30, 2025, 10:51:21 (UTC+05:45)	7.8 KB	Standard
sales.csv	csv	July 30, 2025, 10:51:22 (UTC+05:45)	13.5 KB	Standard

Name	Type	Last modified	Size	Storage class
products.json	json	July 30, 2025, 10:51:43 (UTC+05:45)	14.0 KB	Standard
sales.json	json	July 30, 2025, 10:51:44 (UTC+05:45)	58.2 KB	Standard

Figure 1 S3 Bucket Creation

## Step 2: PostgreSQL & PgAdmin Setup

- Installed PostgreSQL (via installer/Homebrew/apt depending on OS).
- Created a new server group in pgAdmin named FDE.
- Registered PostgreSQL server with username postgres and password set during installation.
- Created a new FDE database with schema landing.

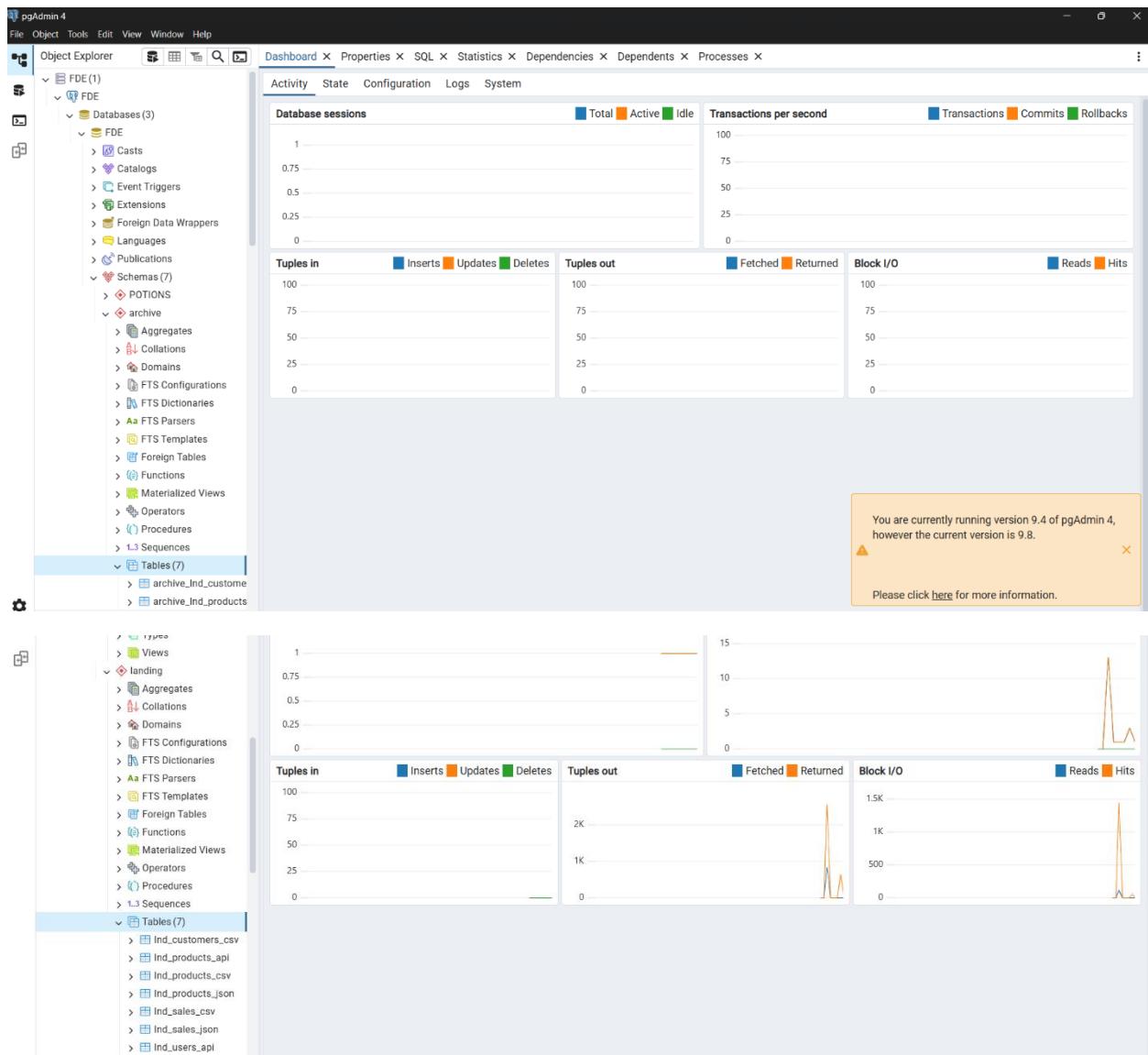


Figure 2 PgAdmin Setup

## Week 2 – Data Extraction (Continued)

### Step 3: Database Connection & Environment Setup

- Created a virtual environment (venv) and activated it.
- Installed dependencies using pip install -r requirements.txt.
- Created database\_connector.py using psycopg2 and SQLAlchemy for DB connections.
- Configured logger for monitoring and debugging.

### Step 4: Extractor Modules

- CSV Extractor: Loaded CSV files from S3 → Landing tables using pandas + SQLAlchemy.
- JSON Extractor: Inserted raw JSON into landing tables using psycopg2.extras.Json.
- Public S3 Extractor: Built get\_public\_url() and extract\_file() to handle file downloads.
- API Extractor: Connected to external APIs, parsed JSON responses, and loaded data.
- Main Extractor: Integrated all extractors, managed configs, handled logging & execution.



A screenshot of a terminal window in a code editor interface. The terminal tab is selected at the top. The command `venv\Scripts\activate` is being typed into the terminal. The output shows the command being run and the prompt changing to `(venv) PS D:\Foundational Data Engineering\FDE>`, indicating that the virtual environment has been successfully activated.

Figure 3 venv activation

FOLDER: FDE

Extractor

- \_pycache\_
- .env
- api\_extractor.py
- archive.py
- config.yaml
- csv\_extractor.py
- database\_connector.py 2
- json\_extractor.py
- main\_extractor.py
- s3\_extractor.py

Loader

- \_pycache\_
- .env
- config.yaml
- products.py
- sales.py
- users.py
- utils.py
- venv

requirements.txt

main\_extractor.py config.yaml archive.py database\_connector.py .env

Extractor > database\_connector.py ...

```
1 import psycopg2
2 from sqlalchemy import create_engine
3 import logging
4
5 logging.basicConfig(level=logging.INFO)
6 logger = logging.getLogger(__name__)
7
8 Windsurf: Refactor | Explain
9 class DatabaseConnector:
10     Windsurf: Refactor | Explain | Generate Docstring | X
11         def __init__(self, config):
12             self.config = config['database']
13
14 Windsurf: Refactor | Explain | Generate Docstring | X
15         def get_connection(self):
16             return psycopg2.connect(**self.config)
17
18 Windsurf: Refactor | Explain | Generate Docstring | X
19         def get_engine(self):
20             conn_str = f"postgresql://[{self.config['user']}]:{self.config['password']}@{self.config['host']}"
21             return create_engine(conn_str)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

PS D:\Foundational Data Engineering\FDE> venv\scripts\activate  
->  
○ (venv) PS D:\Foundational Data Engineering\FDE> [ ]

bash powershell

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF () Python 3.12.4 Go Live AI Code Chat Windsurf (...) Prettier

Folders: FDE

Extractor

- `main_extractor.py`
- ! `config.yaml`
- `archive.py`
- ! `archive.py`
- ! `config.yaml`
- ! `csv_extractor.py`
- `database_connector.py`
- `json_extractor.py`
- `main_extractor.py`
- ! `s3_extractor.py`
- Loader

  - `_pycache_`
  - .env
  - ! `config.yaml`
  - `products.py`
  - `sales.py`
  - `users.py`
  - `utils.py`
  - > venv

requirements.txt

File Edit Selection View Go Run Terminal Help ← → ⚡ FDE

CHAT + ⚡ ... | ⚡

Welcome to Copilot

Let's get started

Add context (#), extensions (@), com

Build workspace

Show project config

Review AI output carefully before use.

```
Extractor > csv_extractor.py > CSVExtractor > load_to_landing
1 import logging
2 import re
3 from datetime import datetime
4
5 logging.basicConfig(level=logging.INFO)
6 logger = logging.getLogger(__name__)
7
8 Windsurf: Refactor | Explain
9 class CSVExtractor:
10     Windsurf: Refactor | Explain | Generate Docstring | X
11     def __init__(self, db_connector):
12         self.db_connector = db_connector
13
14     Windsurf: Refactor | Explain | X
15     @staticmethod
16     def camel_to_snake(name):
17         """
18             Convert camelCase or PascalCase to snake_case.
19             Example: 'customerKey' -> 'customer_key', 'StoreID' -> 'store_id'
20         """
21
22         s1 = re.sub("([a-z])([a-zA-Z]+)", r"\1_\2", name)
23         return re.sub("([a-zA-Z])([a-9])", r"\1_\2", s1).lower()
24
25 Windsurf: Refactor | Explain | Generate Docstring | X
26     def load_to_landing(self, table_name, df, source_info, main_extractor=None):
27         table_name = table_name.lower()
28         table_columns = main_extractor.get_table_columns(table_name, schema="landing")
29         engine = self.db_connector.get_engine()
30         try:
31
32             Windsurf: Refactor | Explain | Generate Docstring | X
33             def normalize_column(col):
34                 col = col.replace(" ", "").replace("-", "_")
35                 col = re.sub(r"-+", "_", col)
36                 return self.camel_to_snake(col)
37
38             df.columns = [normalize_column(col) for col in df.columns]
```

**.env**

```
1 # Database Configuration
2 DB_HOST=localhost
3 DB_USER=postgres
4 DB_PASSWORD=shashank7
5 DB_PORT=5432
6
7
8 # S3 Configuration
9 S3_BUCKET_NAME=shawshank77
10 AWS_REGION=us-east-1
11
```

**main\_extractor.py**

```
1 import logging
2 import os
3 from string import Template
4
5 import yaml
6 from dotenv import load_dotenv
7 from sqlalchemy import text
8
9 from Extractor.api_extractor import APIExtractor
10 from Extractor.csv_extractor import CSVExtractor
11 from Extractor.database_connector import DatabaseConnector
12 from Extractor.json_extractor import JSONExtractor
13 from Extractor.s3_extractor import Publics3Extractor
14
15 load_dotenv()
16
17 logging.basicConfig(level=logging.INFO)
18 logger = logging.getLogger(__name__)
19
20
21 class MainExtractor:
22     def __init__(self, config_path="config.yaml"):
23         self.config = self.load_config(config_path)
24         self.setup_extractors()
25
26     def load_config(self, config_path):
27         with open(config_path, "r") as file:
28             config_content = file.read()
29
30             template = Template(config_content)
31             config_content = template.safe_substitute(os.environ)
32
33         return yaml.safe_load(config_content)
```

Figure 4 Requirement installation and all the setups

File Object Tools Edit View Window Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes landing.Ind\_customers\_csv/FDE/postgres@FDE

Query History

```
1 ✓ SELECT * FROM landing.lnd_customers_csv
2 ORDER BY id ASC
```

Scratch Pad

Data Output Messages Notifications

Showing rows: 1 to 51 Page No: 1 of 1

	<b>id</b> [PK] integer	<b>customer_key</b> character varying (50)	<b>gender</b> character varying (20)	<b>name</b> character varying (255)	<b>city</b> character varying (100)	<b>state_code</b> character varying (50)	<b>state</b> character varying (100)	<b>zip_code</b> character varying (20)
1	1	CUST001	Female	Sarah Johnson	New York	NY	New York	10001
2	2	CUST002	Male	Michael Chen	Los Angeles	CA	California	90001
3	3	CUST003	Non-binary	Alex Rivera	Chicago	IL	Illinois	60601
4	4	CUST004	Female	Jessica Kim	Seattle	WA	Washington	98101
5	5	CUST005	Male	David Wilson	Miami	FL	Florida	33101
6	6	CUST006	Female	Emily Davis	Boston	MA	Massachusetts	2108
7	7	CUST007	Male	James Miller	Austin	TX	Texas	73301
8	8	CUST008	Non-binary	Taylor Smith	Denver	CO	Colorado	80201
9	9	CUST009	Female	Amanda Brown	Atlanta	GA	Georgia	30301
10	10	CUST010	Male	Daniel Lee	Portland	OR	Oregon	97201
11	11	CUST011	Female	Michelle Garcia	San Francisco			
12	12	CUST012	Male	Robert Martinez	Philadelphia	PA	Pennsylvania	19101
13	13	CUST013	Male	John Doe	Chicago	IL	Illinois	60601
14	14	CUST014	Female	Jane Smith	Los Angeles	CA	California	90001
15	15	CUST015	Male	Mike Johnson	Seattle	WA	Washington	98101

Successfully run. Total query runtime: 348 msec. 51 rows affected.

*Figure 5 Loaded data in landing table*

# Week 3- Created all the Schemas left (Staging, Target, Transform)

## Step 5: Schemas

- Created all the remaining Schemas i.e. Staging, Target, and Transform
- Loaded the data to the tables

The screenshot shows the pgAdmin 4 interface. The Object Explorer on the left lists various database objects under the 'landing' schema. A query window in the center displays the following SQL code and its results:

```

SELECT * FROM landing.lnd_customers_csv
ORDER BY id ASC
  
```

The results table shows 51 rows of customer data:

id	customer_key	gender	name	city	state_code	state	zip_code
1	CUST001	Female	Sarah Johnson	New York	NY	New York	10001
2	CUST002	Male	Michael Chen	Los Angeles	CA	California	90001
3	CUST003	Non-binary	Alex Rivera	Chicago	IL	Illinois	60601
4	CUST004	Female	Jessica Kim	Seattle	WA	Washington	98101
5	CUST005	Male	David Wilson	Miami	FL	Florida	33101
6	CUST006	Female	Emily Davis	Boston	MA	Massachusetts	2108
7	CUST007	Male	James Miller	Austin	TX	Texas	73301
8	CUST008	Non-binary	Taylor Smith	Denver	CO	Colorado	80201
9	CUST009	Female	Amanda Brown	Atlanta	GA	Georgia	30301
10	CUST010	Male	Daniel Lee	Portland	OR	Oregon	97201
11	CUST011	Female	Michelle Garcia	San Francisco	CA	California	94101
12	CUST012	Male	Robert Martinez	Philadelphia	PA	Pennsylvania	19101

Total rows: 51 Query complete 00:00:00.348

Figure 6 Created the schemas (Staging, Transform, and Target)

The screenshot shows the pgAdmin 4 interface. The Object Explorer on the left lists various database objects under the 'target' schema. A query window in the center displays the following SQL code and its results:

```

SELECT * FROM target.dim_products
ORDER BY product_id ASC
  
```

The results table shows 155 rows of product data:

product_id	product_name	category	brand	price	stock_quantity	sku	so	ch
PRD001	Essence Mascara Lash Princess	beauty	Essence	9.99	99	BEA-ESS-ESS-001	AI	
PRD002	Eyeshadow Palette with Mirror	beauty	Glamour Beauty	19.99	34	BEA-GLA-EYE-002	AI	
PRD003	Powder Canister	beauty	Velvet Touch	14.99	89	BEA-VEL-POW-003	AI	
PRD004	Red Lipstick	beauty	Chic Cosmetics	12.99	91	BEA-CHI-LIP-004	AI	
PRD005	Red Nail Polish	beauty	Nail Couture	8.99	79	BEA-NAI-NAI-005	AI	
PRD006	Calvin Klein CK One	fragrances	Calvin Klein	49.99	29	FRA-CAL-CAL-006	AI	
PRD007	Chanel Coco Noir Eau De	fragrances	Chanel	129.99	58	FRA-CHA-CHA-007	AI	
PRD008	Dior J'adore	fragrances	Dior	89.99	98	FRA-DIO-DIO-008	AI	
PRD009	Dolce Shine Eau de	fragrances	Dolce & Gabbana	69.99	4	FRA-DOL-DOL-009	AI	
PRD010	Gucci Bloom Eau de	fragrances	Gucci	79.99	91	FRA-GUC-GUC-010	AI	
PRD011	Annibale Colombo Bed	furniture	Annibale	2499.99	60	FUR-ANN-ANN-012	AI	
PRD012	Annibale Colombo Sofa	furniture	Annibale Colombo	2499.99	60	FUR-ANN-ANN-012	AI	

Successfully run. Total query runtime: 169 msec. 155 rows affected.

FDE/FDE - Database connected

Figure 7 Loaded the data

## Week 4- Data Loading

### Step 6: Loader Setup

- Created a Loader folder for loading scripts.
- Configured .env file with PostgreSQL credentials.
- Created config.yaml with mappings of schemas (landing, staging, transform, target) and entities (products, users, sales).

### Step 7: Utilities (utils.py)

- Implemented load\_config(), get\_db\_connection(), get\_schemas(), get\_entities(), and execute\_query() for DB + config handling.

### Step 8: Loading Dimension Tables (products, users)

- Cleared temp tables before each run.
- Inserted required columns from staging views → temp tables.
- Used MERGE (UPSERT) with ON CONFLICT to insert/update target tables.
- Added updated\_at timestamp for tracking changes.

### Step 9: Loading Fact Table (sales)

- Cleaned missing store\_id values (verified with data provider).
- Handled missing unit\_price by filling from product table.
- Calculated total\_amount = quantity × unit\_price.
- Applied deduplication using DISTINCT ON to prevent duplicate sales.

### Step 10: Script Execution

- Ran dimension load scripts first (products, users).
- Then ran fact\_sales load script (ensuring foreign key references exist).
- Verified loaded data in pgAdmin target tables.

The screenshot shows a code editor interface with a sidebar on the left containing icons for file operations like Open, Save, Find, Replace, and a refresh button. The main area displays a file tree and a code editor.

**File Tree:**

- ✓ s3\_extractor.py
- ✓ Loader
  - > \_\_pycache\_\_
  - ⚙ .env
  - ✗ config.yaml
  - ✓ products.py
  - ✓ sales.py
  - ✓ users.py
  - ✓ utils.py
  - > venv
- ☰ requirements.txt

**Code Editor (s3\_extractor.py):**

```
1 Windsurf: Refactor
2 class Publics:
3     def __in:
4         self.
5         self.
6         self.
7         self.
8         self.
9         self.
10        self.
11        self.
12    def get_1
13    # Com
14    return
```

*Figure 8 Loader folder created*

The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it displays the project structure under "FOLDERS: FDE". The "Extractor" folder contains files like `api_extractor.py`, `archive.py`, `config.yaml`, `csv_extractor.py`, `database_connector.py`, `json_extractor.py`, `main_extractor.py`, and `s3_extractor.py`. The "Loader" folder contains files like `products.py`, `sales.py`, `users.py`, and `utils.py`. A `venv` folder and a `requirements.txt` file are also listed.
- Editor Area:** The main area shows the content of the `.env` file. The file contains environment variables for database and AWS configurations:

```
1 # Database Configuration
2 DB_HOST=localhost
3 DB_USER=postgres
4 DB_PASSWORD=shashank77
5 DB_PORT=5432
6
7
8 # S3 Configuration
9 S3_BUCKET_NAME=shashank77
10 AWS_REGION=us-east-1
11
```
- Toolbar:** At the top, there are standard file operations (File, Edit, Selection, View, Go, Run, Terminal, Help) and a search bar.
- Status Bar:** The bottom status bar shows "FDE" and some icons.
- Right Sidebar:** A "CHAT" section with a "Welcome to Copilot" message and a "Let's get started" button. It also includes buttons for "Build workspace" and "Show project config". A note at the bottom says "Review AI output carefully before use."

*Figure 9 Setup .env*

```

FOLDERS: FDE
Extractor
    > __pycache__
    .env
    api_extractor.py
    archive.py
    config.yaml
    csv_extractor.py
    database_connector.py
    json_extractor.py
    main_extractor.py
    s3_extractor.py
Loader
    > __pycache__
    .env
    config.yaml
    products.py
    sales.py
    users.py
    utils.py
    venv
requirements.txt

config.yaml
1 database:
2   host: ${DB_HOST}
3   database: FDE
4   user: ${DB_USER}
5   password: ${DB_PASSWORD}
6   port: ${DB_PORT}
7
8 schemas:
9   landing_schema: landing
10  staging_schema: staging
11  transform_schema: transform
12  target_schema: target
13
14
15 entities:
16   products:
17     staging_view: stg_products
18     temp_table: tmp_products
19     target_table: dim_products
20   users:
21     staging_view: stg_users
22     temp_table: tmp_users
23     target_table: dim_users
24   sales:
25     staging_view: stg_sales
26     temp_table: tmp_sales
27     target_table: fact_sales
28

```

Figure 10 Setup config.yaml

```

FOLDERS: FDE
Extractor
    > __pycache__
    .env
    api_extractor.py
    archive.py
    config.yaml
    csv_extractor.py
    database_connector.py
    json_extractor.py
    main_extractor.py
    s3_extractor.py
Loader
    > __pycache__
    .env
    config.yaml
    products.py
    sales.py
    users.py
    utils.py
    venv
requirements.txt

products.py
1 from psycopg2 import sql
2 from Loader.utils import get_db_connection, execute_query, get_schemas, get_entities
3
4 ENTITY = 'products'
5
6 WindSurf Refactor | Explain | X
7 def load_products():
8     """Load products data from staging to target"""
9     schemas = get_schemas()
10    entities = get_entities()
11
12    conn = get_db_connection()
13    try:
14        # Step 1: Clear temp table
15        execute_query(conn, f"TRUNCATE TABLE {schemas['transform_schema']}.{entities[ENTITY]['temp_table']}")
16
17        # Step 2: Load staging data to temp table
18        load_query = sql.SQL("""
19            INSERT INTO {transform_table} (
20                product_id, product_name, category, brand, price,
21                stock_quantity, sku, source_system, source_loaded_at
22            )
23            SELECT
24                product_id, product_name, category, brand, price,
25                stock_quantity, sku, source_system, source_loaded_at
26            FROM {staging_view}
27        """).format(
28            transform_table=sql.Identifier(schemas['transform_schema']), entities[ENTITY]['temp_table']
29            staging_view=sql.Identifier(schemas['staging_schema']), entities[ENTITY]['staging_view']
30        )
31
32        execute_query(conn, load_query)
33
34        # Step 3: Merge into target table
35        merge_query = sql.SQL("""
36            INSERT INTO {target_table} (
37                product_id, product_name, category, brand, price,
38                stock_quantity, sku, source_system
39            )
40        """).format(
41            target_table=sql.Identifier(schemas['target_schema']), entities[ENTITY]['target_table']
42        )
43
44        execute_query(conn, merge_query)
45
46    except Exception as e:
47        print(f"Error during product load: {e}")
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
236
237
237
238
239
239
240
241
242
243
244
245
245
246
247
247
248
249
249
250
251
252
253
254
255
255
256
257
257
258
259
259
260
261
262
263
264
265
265
266
267
267
268
269
269
270
271
272
273
274
275
275
276
277
277
278
279
279
280
281
282
283
284
285
285
286
287
287
288
289
289
290
291
292
293
294
295
295
296
297
297
298
299
299
300
301
302
303
304
305
305
306
307
307
308
309
309
310
311
312
313
314
314
315
316
316
317
318
318
319
319
320
321
322
323
324
324
325
326
326
327
328
328
329
329
330
331
332
333
333
334
335
335
336
336
337
337
338
338
339
339
340
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
137
```

The screenshot shows a Python IDE interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Find, as well as a refresh icon.
- Sidebar:** Shows a tree view of the project structure:
  - Extractor:** Contains .env, api\_extractor.py, archive.py, config.yaml, csv\_extractor.py, database\_connector.py, json\_extractor.py, main\_extractor.py (selected), and s3\_extractor.py.
  - Loader:** Contains .env, config.yaml, products.py, sales.py (selected), users.py, and utils.py.
  - venv**
  - requirements.txt
- Code Editor:** Displays the content of `sales.py`. The code implements a three-step ETL process:
  - Step 1:** Clear temp table.
  - Step 2:** Load staging data to temp table. This involves an `INSERT INTO` query with columns: sale\_id, sale\_date, store\_id, product\_id, customer\_id, quantity, unit\_price, total\_amount, source\_system, and source\_loaded\_at. It also includes a `SELECT` statement for the same columns from a `{staging_view}`.
  - Step 3:** Fill missing unit\_price from product dimension. This involves an `UPDATE` query on the `{transform_table}` table, setting `unit_price` to `p.price` where `p` is from the `{product_table}`.
- Status Bar:** Shows "FDE" and a refresh icon.

The screenshot shows a VS Code interface with the following details:

- File Explorer:** On the left, it lists files and folders. The 'users.py' file under the 'Loader' folder is currently selected.
- Editor:** The main editor area contains the code for 'main\_extractor.py'. It includes imports for psycopg2, Loader, and various utility functions. The code implements a three-step ETL process: 1) Clearing a temp table, 2) Loading staging data into the temp table, and 3) Merging the temp table into the target table.
- Terminal:** At the bottom, there is a terminal window showing the output of the command 'git status'.
- Copilot Panel:** On the right, there is a 'CHAT' panel titled 'Welcome to Copilot'. It says 'Let's get started' and has buttons for 'Add context (#), extensions (@), com' (disabled), 'Build workspace', and 'Show project config'.

```

FOLDERS: FDE
Extractor
    _pycache_
    .env
    api_extractor.py
    archive.py
    config.yaml
    csv_extractor.py
    database_connector.py
    json_extractor.py
    main_extractor.py
    s3_extractor.py
Loader
    _pycache_
    .env
    config.yaml
    products.py
    sales.py
    users.py
    utils.py
utils.py
requirements.txt

main_extractor.py 2
utils.py 2
archive.py 1
.env

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

```

Figure 11 All python files

user_id	first_name	last_name	email	phone	age	gender	city
CUS001	Emily	Johnson	emily.johnson@x.dummyjson.com	+81 965-431-3024	28	female	Phoenix
CUS002	Michael	Williams	michael.williams@x.dummyjson.com	+49 256-627-6644	35	male	Houston
CUS003	Sophia	Brown	sophia.brown@x.dummyjson.com	+81 210-652-2785	42	female	Washington
CUS004	James	Davis	james.davis@x.dummyjson.com	+49 614-958-9364	45	male	Seattle
CUS005	Emma	Miller	emma.miller@x.dummyjson.com	+91 759-776-1614	30	female	Jackson
CUS006	Olivia	Wilson	olivia.wilson@x.dummyjson.com	+91 607-295-6448	22	female	Fort Worth
CUS007	Alexander	Jones	alexander.jones@x.dummyjson.com	+61 260-824-4986	38	male	Indiana
CUS008	Ava	Taylor	ava.taylor@x.dummyjson.com	+1 458-853-7877	27	female	Fort Worth
CUS009	Ethan	Martinez	ethan.martinez@x.dummyjson.com	+92 933-608-5081	33	male	San Antonio
CUS010	Isabella	Anderson	isabella.anderson@x.dummyjson.com	+49 770-658-4885	31	female	New York
CUS011	Liam	Garcia	liam.garcia@x.dummyjson.com	✓ Successfully run. Total query runtime: 110 msec. 81 rows affected.	24	female	Jackson
CUS012	Mia	Rodriguez	mia.rodriguez@x.dummyjson.com	+49 989-461-8403	26	female	Los Angeles

Figure 12 Data after loading

## Week 5 – Orchestration with Prefect

### Step 11: Orchestration Script (pipeline.py)

- Created orchestration/pipeline.py.
- Imported extract, load, archive tasks from Extractor & Loader modules.
- Configured task retries (3 retries with 30s delay).
- Designed flow sequence: Extract → Load (users & products in parallel) → Load sales → Archive.

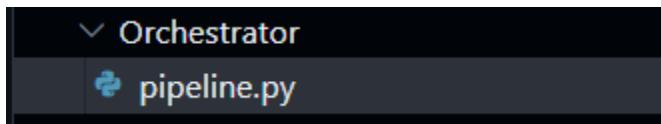


Figure 13 Created Orchestrator and the file

### Step 12: Running the Pipeline Locally

- Started Prefect Orion server using:  
→ prefect server start
- Set environment variable PREFECT\_API\_URL.
- Ran pipeline using:  
→ python orchestrator/pipeline.py
- Verified execution in Prefect Dashboard (<http://127.0.0.1:4200/dashboard>).

### Step 13: Scheduling the Pipeline

- Deployed pipeline with cron job using:  
→ prefect deploy orchestrator/pipeline.py:fde\_pipeline --name "fde-pipeline" --cron "13 15 \* \* \*"
- Created a work pool in Prefect.
- Started worker with:  
→ prefect worker start --pool 'FDE'
- Verified scheduled deployment in Prefect Deployment tab.
- Observed successful scheduled pipeline run at specified time.

```
File Edit Selection View Go Run Terminal Help < > FDE
FOLDERS: FDE
Extractor
> __pycache__
__env
api_extractor.py
archive.py
config.yaml
csv_extractor.py
database_connector.py
json_extractor.py
main_extractor.py
s3_extractor.py
Loader
> __pycache__
Orchestrator
> __pycache__
Pipeline.py
utils.py
venv
requirements.txt
main_extractor.py 2 config.yaml 2 utils.py 2 pipeline.py 1 archive.py env ... CHAT + ⚡ ... | 🔍 ...
1 import sys
2 from pathlib import Path
3 sys.path.append(str(Path(__file__).parent.parent))
4
5 from prefect import flow, task
6
7 from Extractor.main_extractor import MainExtractor
8 from Loader.products import load_products
9 from Loader.sales import load_sales
10 from Loader.users import load_users
11 from Extractor.archive import main as archive_main
12
13 Windsurf:Refactor | Explain | Generate Docstring | X
14 @task(retries=3, retry_delay_seconds=60)
15 def extract_task():
16     extractor = MainExtractor()
17     extractor.extract_all()
18
19 Windsurf:Refactor | Explain | Generate Docstring | X
20 @task(retries=2, retry_delay_seconds=30)
21 def load_products_task():
22     load_products()
23
24 Windsurf:Refactor | Explain | Generate Docstring | X
25 @task(retries=2, retry_delay_seconds=30)
26 def load_sales_task():
27     load_sales()
28
29 Windsurf:Refactor | Explain | Generate Docstring | X
30 @task(retries=2, retry_delay_seconds=30)
31 def load_users_task():
32     load_users()
33
34 Windsurf:Refactor | Explain | Generate Docstring | X
35 @task(retries=2, retry_delay_seconds=30)
36 def archive_task():
37     archive_main()
38
```

Welcome to Copilot  
Let's get started  
Add context (#), extensions (@), com  
Build workspace  
Show project config  
Review AI output carefully before use.

Figure 14 Pipeline.py code

```
File Edit Selection View Go Run Terminal PORTS AZURE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
+ ... | < bash
prefect
bash
Add context (#), extensions (@), com
Build workspace
Show project config
Review AI output carefully before use.

PS D:\Foundational Data Engineering\FDE> prefect server start
Switched to profile 'local'

[ - ] [ - ] [ - ] [ - ] [ - ]
[ - ] [ - ] [ - ] [ - ] [ - ]
[ - ] [ - ] [ - ] [ - ] [ - ]
[ - ] [ - ] [ - ] [ - ] [ - ]
[ - ] [ - ] [ - ] [ - ] [ - ]
Configure Prefect to communicate with the server with:
prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api
View the API reference documentation at http://127.0.0.1:4200/docs
Check out the dashboard at http://127.0.0.1:4200
```

Figure 15 Prefect server start

The screenshot shows a dark-themed code editor interface. At the top, there is a navigation bar with icons for File, Edit, Selection, View, Go, Run, and Terminal. Below the navigation bar is a file tree view.

**FOLDERS: FDE**

- Extractor
  - \_pycache\_
  - .env
  - api\_extractor.py
  - archive.py
  - config.yaml
  - csv\_extractor.py
  - database\_connector.py
  - json\_extractor.py
  - main\_extractor.py
  - s3\_extractor.py
- Loader
  - \_pycache\_
- Orchestrator
  - pipeline.py
  - .env
  - config.yaml
  - products.py
  - sales.py
  - users.py
  - utils.py
- venv
- requirements.txt

On the left side of the interface, there is a vertical toolbar with various icons corresponding to common development tasks like search, replace, find, and run. A blue circle with the number "1" is visible near the bottom of this toolbar.

The status bar at the bottom right indicates "1" file is open.

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE + × ⋮ | [ ] X

```
PS D:\Foundational Data Engineering\FDE> venv\Scripts\activate
● >>
● (venv) PS D:\Foundational Data Engineering\FDE> set PREFECT_API_URL=http://127.0.0.1:4200/api
● (venv) PS D:\Foundational Data Engineering\FDE> $env:PREFECT_API_URL="http://127.0.0.1:4200/api"
● (venv) PS D:\Foundational Data Engineering\FDE> set PREFECT_API_URL=http://127.0.0.1:4200/api
○ (venv) PS D:\Foundational Data Engineering\FDE> python orchestrator/pipeline.py
19:41:45.571 | INFO    | Flow run 'roaring-angelfish' - Beginning flow run 'roaring-angelfish' for flow 'FDE Data Pipeline'
19:41:45.577 | INFO    | Flow run 'roaring-angelfish' - View at http://127.0.0.1:4200/runs/flow-run/2f31045b-165b-4a51-9fab-fac988029c97
19:41:45.836 | INFO    | Task run 'extract_task-dcb' - Task run failed with exception: FileNotFoundError(2, 'No such file or directory') - Retry 1/3 will start 60 second(s) from now
```

*Figure 16 Setting the prefect and then executing*

**Runs / carmine-cormorant**

Flow FDE Data Pipeline

Logs Task Runs Subflow Runs Artifacts Details Parameters Job Variables

Ready to scale? Upgrade

Join the Community

Settings

Logs Task Runs Subflow Runs

Ready to scale? Upgrade

Join the Community

Settings

Sep 6th, 2025

INFO Beginning flow run 'carmine-cormorant' for flow 'FDE Data Pipeline'

INFO Task run failed with exception: FileNotFoundError(2, 'No such file or directory') - Retry 1/3 will start 60 second(s) from now

Sep 6th, 2025

INFO Beginning Flow run 'roaring-angelfish' for Flow 'FDE Data Pipeline'

INFO Task run failed with exception: FileNotFoundError(2, 'No such file or directory') - Retry 1/3 will start 60 second(s) from now

Join the Prefect Community

Connect with 25k+ engineers scaling Python with Prefect. Show us your work and be the first to know about new Prefect features.

Join us on Slack

Notify me about Prefect updates

hello@prefect.io

Skip Sign up

Cancel

Level: all Oldest to newest

*Figure 17 Dashboard*

The screenshot displays a developer's environment across three main sections: a code editor, a terminal, and a cloud-based deployment interface.

**Code Editor:** The left section shows a file tree for a project named "FDE". The "Extractor" directory contains files like `config.yaml`, `api\_extractor.py`, `archive.py`, and `main\_extractor.py`. The "pipeline.py" file is open in the terminal tab, showing a cron job definition:

```
pipeline" --cron "13 15 * * *"
>>>
Unable to read the specified config file. Reason: [Errno 2] No such file or directory: 'prefect.yaml'. skipping.
? Looks like you don't have any work pools this flow can be deployed to. Would you like to create one? [y/n] (y):
y ? What infrastructure type would you like to use for your new work pool? [Use arrows to move; enter to select]
```

**Terminal:** The middle section shows the output of running the pipeline script. It indicates that it cannot find the `prefect.yaml` configuration file and asks if the user wants to create a new work pool. The user has selected "prefect" as the infrastructure type.

**Prefect Cloud Deployment View:** The right section shows the "Deployments" page in Prefect Cloud. A single deployment named "fde-pipeline" is listed under the "fde Data Pipeline" flow. The status is "Not Ready". The deployment is scheduled to run at 03:13 PM every day. The deployment table includes columns for Deployment, Status, Activity, Tags, and Schedules.

Figure 18 Before not ready

The screenshot shows the FDE interface with the 'TERMINAL' tab selected. The terminal window displays the command: `(venv) PS D:\Foundational Data Engineering\FDE> prefect worker start --pool 'FDE'`. Below the terminal is a 'Deployments' section. On the left, a sidebar lists various FDE components: Dashboard, Runs, Flows, Deployments (which is selected), Work Pools, Blocks, Variables, Automations, Event Feed, and Concurrency. The main area shows a table titled 'Deployments' with one entry: 'fde-pipeline' (Status: Ready, Activity: .....). A search bar at the top right allows filtering by 'Search deployments...', 'All tags', and sorting by 'A to Z'. Below the table are buttons for 'Items per page' (set to 10), navigation arrows, and a page indicator 'Page 1 of 1'. At the bottom of the interface, there are links for 'Ready to scale?', 'Upgrade', 'Join the Community', and 'Settings'.

Figure 19 After the deployment

This screenshot shows the 'Deployments / fde-pipeline' page. The left sidebar remains the same as in Figure 19. The main content area is titled 'Deployments / fde-pipeline' and shows the 'fde-pipeline' deployment details. It includes tabs for 'Runs', 'Upcoming', 'Parameters', 'Configuration', and 'Description'. The 'Runs' tab is active, showing a single run entry: 'FDE Data Pipeline > expert-raven' with status 'Scheduled' and scheduled for '2025/09/06 08:58:00 PM'. The 'Description' tab is also visible. On the right side, there are sections for 'Schedules' (with a green toggle switch for 'At 03:13 PM every day'), 'Triggers' (with a '+ Add' button), and 'Status' (showing 'Ready'). Below these are detailed logs and deployment metadata. At the bottom, there are links for 'Ready to scale?', 'Upgrade', 'Join the Community', and 'Settings'.

## Week 6 – Visualization (Metabase)

### Step 14: Metabase Setup

- Installed and configured Metabase.
- Connected PostgreSQL (FDE database) as a data source.

### Step 15: Dashboard Creation

- Created charts and visualizations:
- Waste requests by category
- Daily/weekly task completion trend
- Total processed waste vs pending
- Top contributors (citizens/organizations)

### Step 16: Final Dashboard

- Combined charts into a Metabase dashboard.
- Took screenshots for submission.

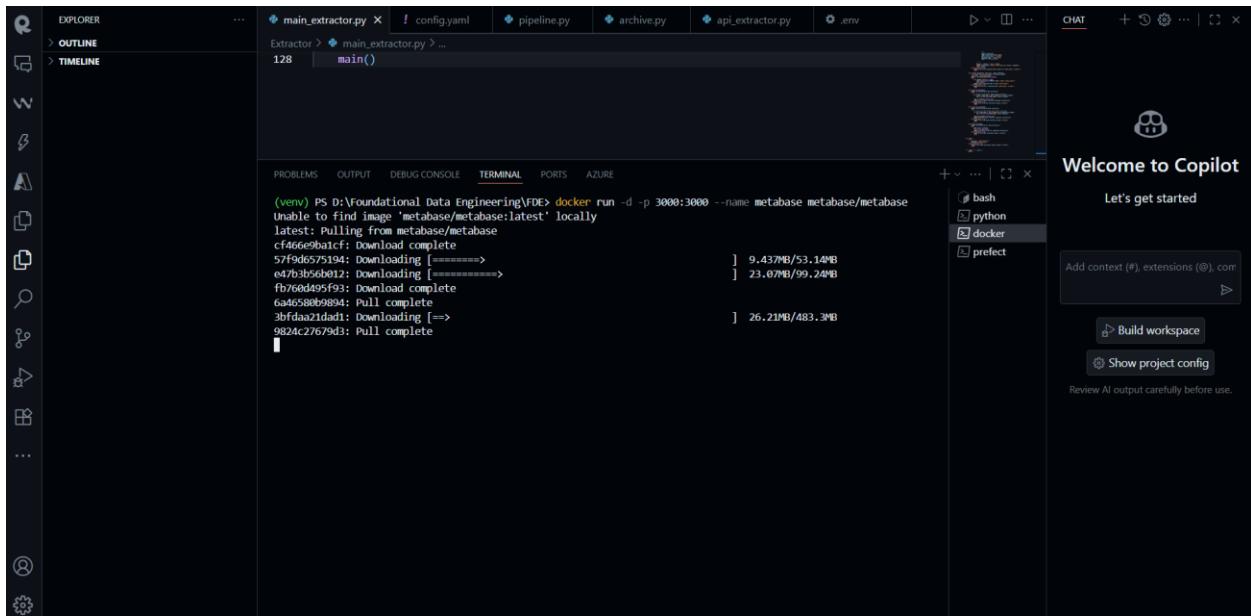


Figure 20 Setting the docker

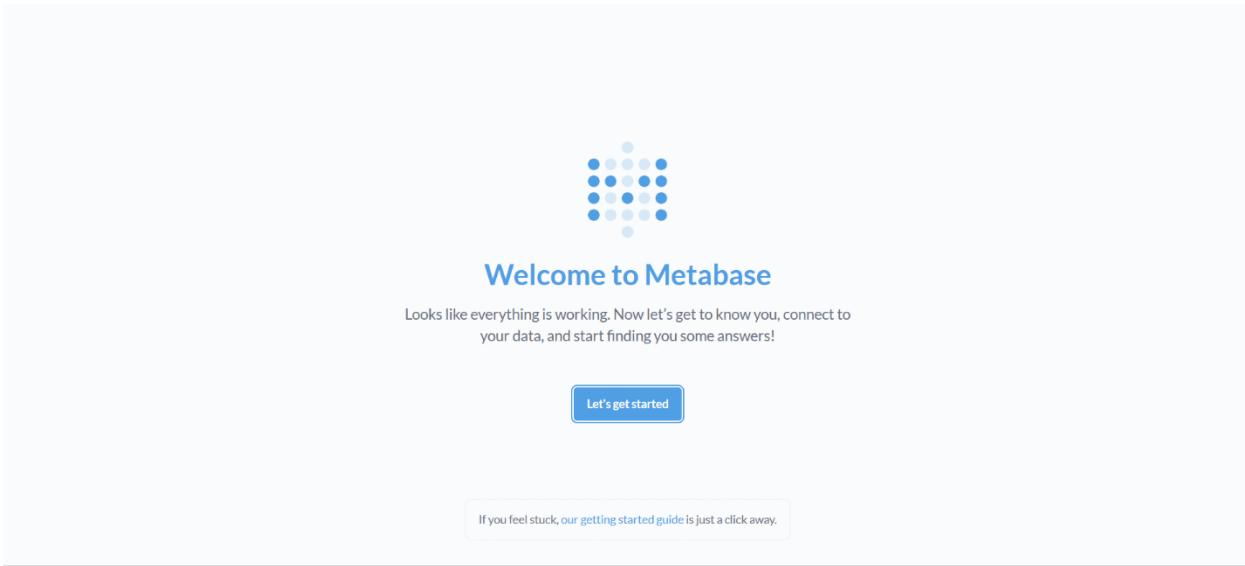


Figure 21 Metabase dashboard

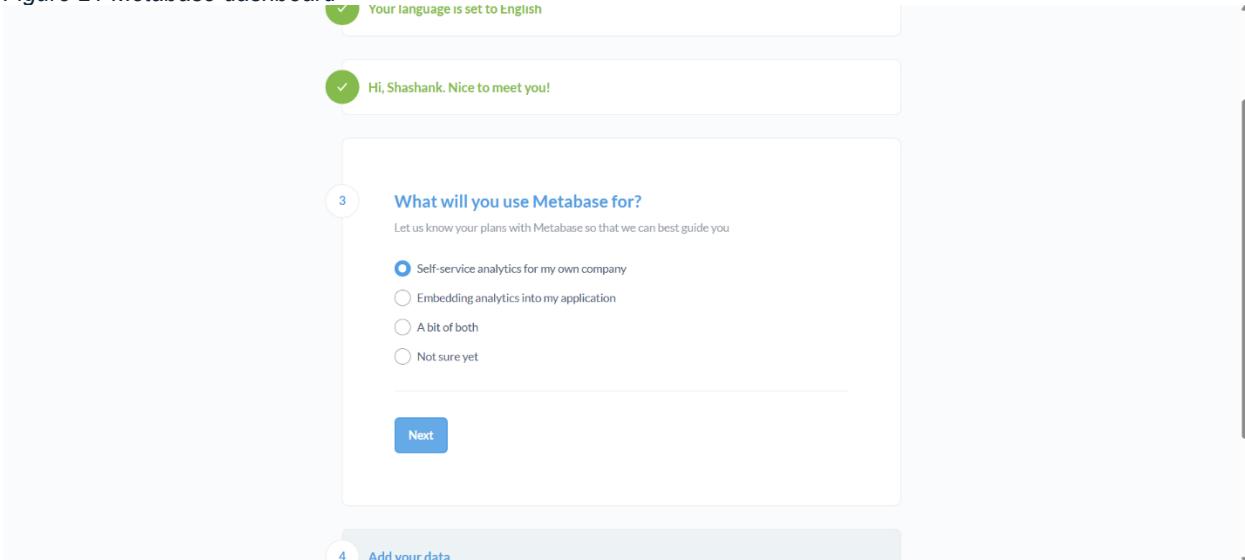
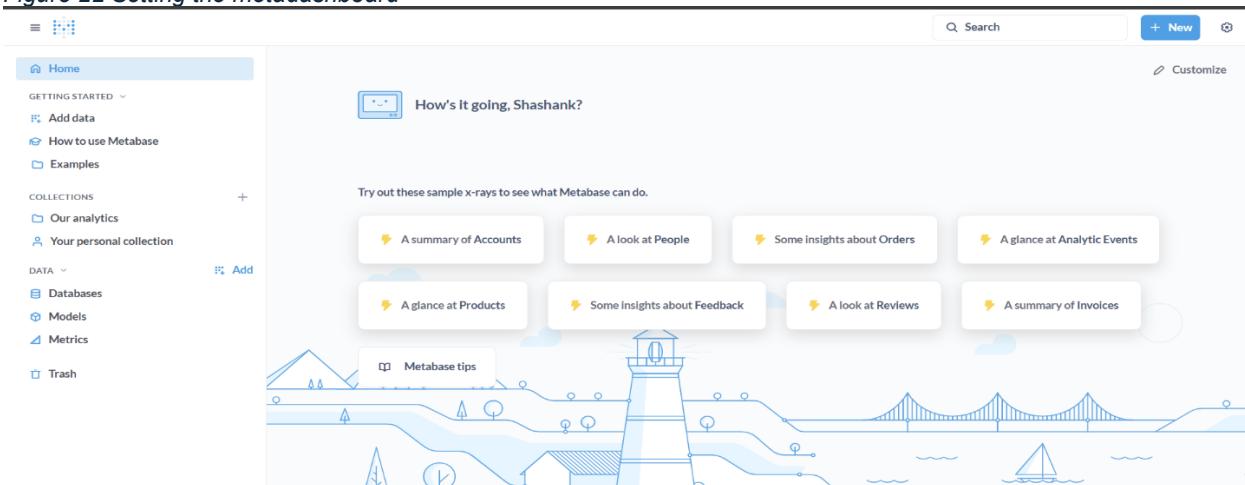


Figure 22 Setting the metashboard



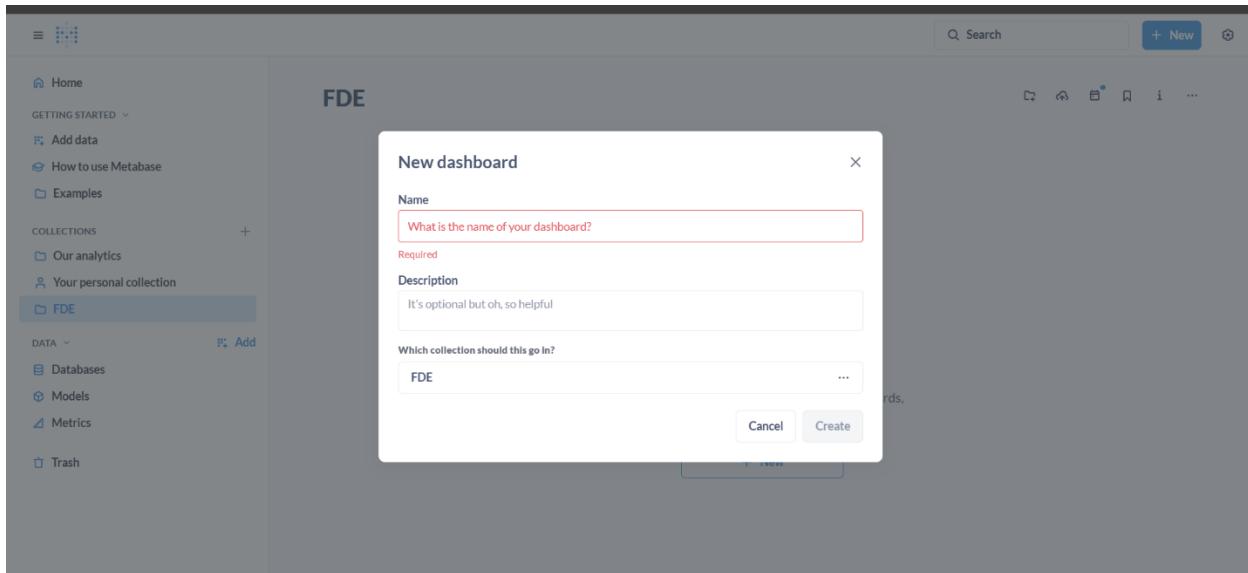
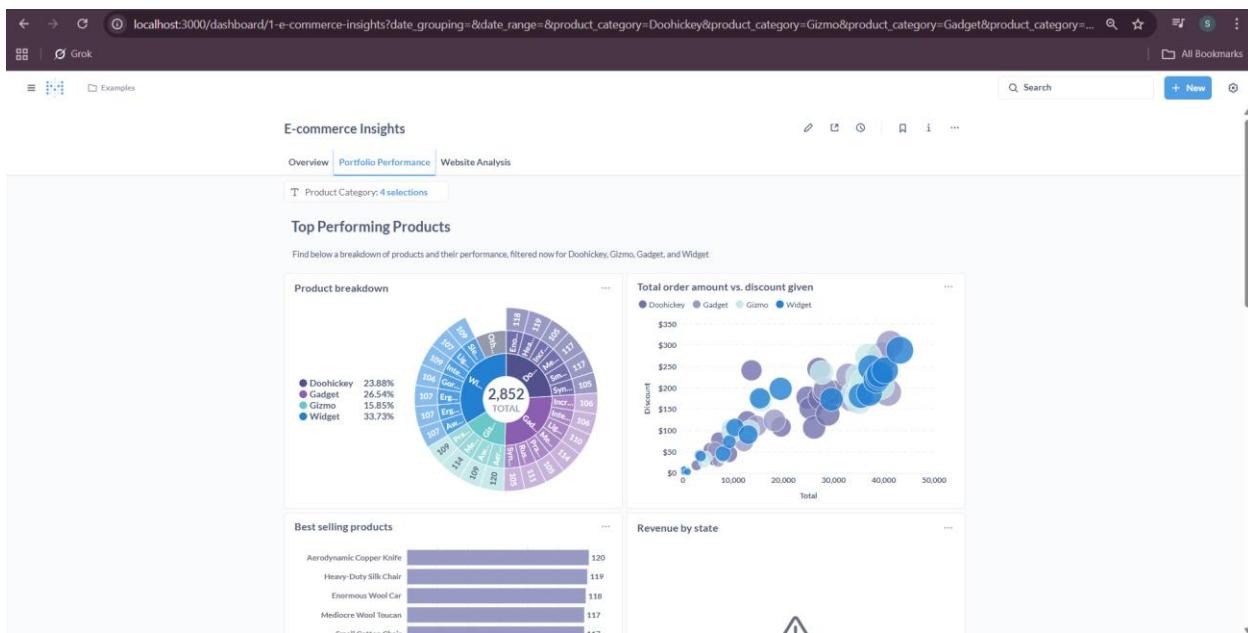


Figure 23 New dashboard for FDE



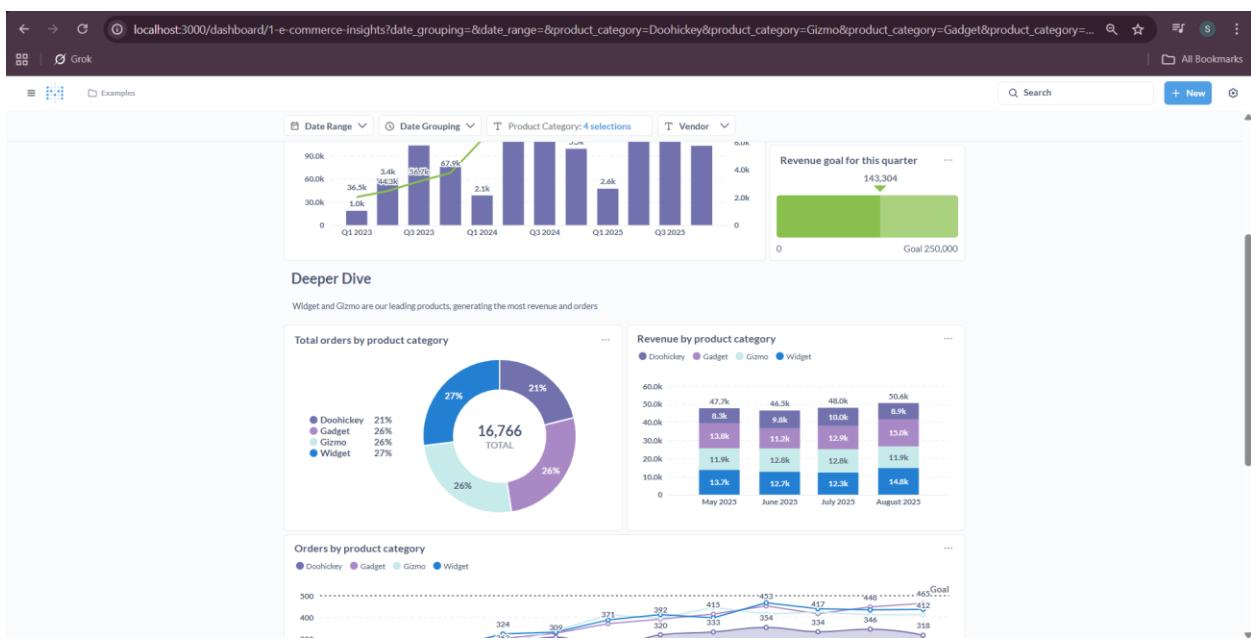
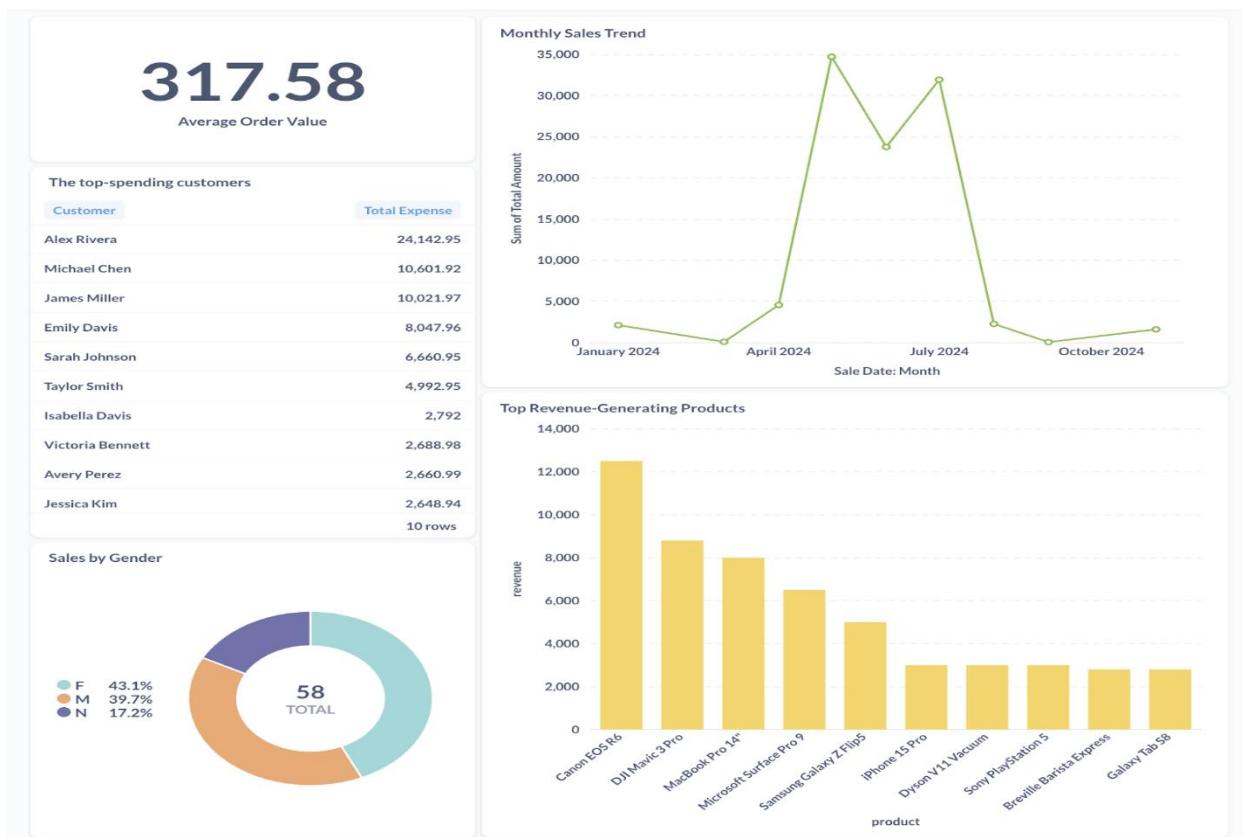


Figure 24 After completion, the graph of the table

## Conclusion

The stepwise implementation of the Foundational Data Engineering (FDE) pipeline provided comprehensive learning experience in building an end-to-end data workflow. Beginning with the setup of AWS S3 for raw data ingestion and PostgreSQL as a data warehouse, the project systematically progressed through extraction, transformation, loading, orchestration, and visualization.

In Week 1 and 2, the focus was on establishing the environment, configuring PostgreSQL, and developing extractor modules for CSV, JSON, S3, and API data. This ensured that diverse data sources could be ingested reliably into the landing schema. In Week 4, the emphasis shifted to loading processes, where dimension and fact tables were carefully designed with utilities for data quality checks, deduplication, and UPSERT operations. This stage ensured that data was both accurate and analysis ready.

Week 5 introduced orchestration with Prefect, which automated the ETL pipeline, implemented retries for resilience, and scheduled flows for periodic execution. This eliminated manual intervention and aligned the pipeline with industry-standard practices. Finally, in Week 6, the pipeline output was connected to Metabase for visualization. The resulting dashboards transformed raw data into actionable insights, enabling monitoring of trends, performance, and decision-making metrics.

Overall, this project demonstrated how a well-structured data engineering pipeline can transform raw, heterogeneous data into a clean, reliable, and continuously updated analytical resource. Beyond fulfilling the module's objectives, the experience has built a practical understanding of modern data engineering tools such as PostgreSQL, Python (pandas, SQLAlchemy, psycopg2), Prefect, and Metabase. The project reflects not only technical implementation but also the importance of automation, data quality assurance, and visualization in enabling organizations to derive value from data.