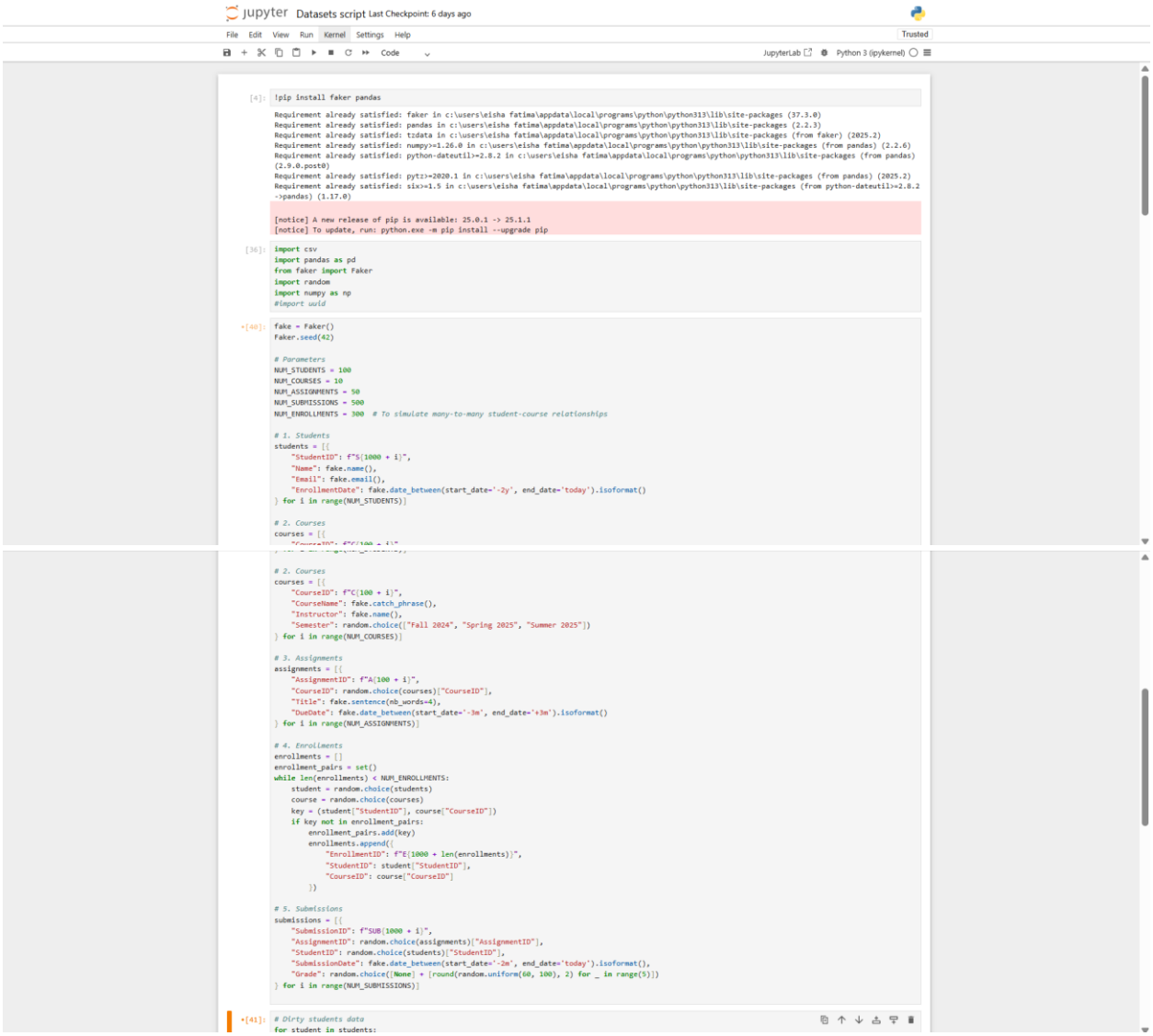
Screenshots for Solution

Script for simulating datasets:



jupyter Datasets script Last Checkpoint: 6 days ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[4]:
```
!pip install faker pandas
```

```
Requirement already satisfied: faker in c:\users\eisha fatima\appdata\local\programs\python\python313\lib\site-packages (37.3.0)
Requirement already satisfied: pandas in c:\users\eisha fatima\appdata\local\programs\python\python313\lib\site-packages (2.2.3)
Requirement already satisfied: tzdata in c:\users\eisha fatima\appdata\local\programs\python\python313\lib\site-packages (from faker) (2025.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\eisha fatima\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.2.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\eisha fatima\appdata\local\programs\python\python313\lib\site-packages (from pandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\eisha fatima\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\eisha fatima\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.8.2
->pandas) (1.17.0)

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[36]:
```
import csv
import pandas as pd
from faker import Faker
import random
import numpy as np
#import uuid
```

*[40]:
```
fake = Faker()
Faker.seed(42)

# Parameters
NUM_STUDENTS = 100
NUM_COURSES = 10
NUM_ASSIGNMENTS = 50
NUM_SUBMISSIONS = 500
NUM_ENROLLMENTS = 300  # To simulate many-to-many student-course relationships

# 1. Students
students = [{
    "StudentID": f"S{1000 + i}",
    "Name": fake.name(),
    "Email": fake.email(),
    "EnrollmentDate": fake.date_between(start_date='-2y', end_date='today').isoformat()
} for i in range(NUM_STUDENTS)]

# 2. Courses
courses = [{
    "CourseID": f"C{100 + i}",
    "CourseName": fake.catch_phrase(),
    "Instructor": fake.name(),
    "Semester": random.choice(["Fall 2024", "Spring 2025", "Summer 2025"])
} for i in range(NUM_COURSES)]

# 3. Assignments
assignments = [{
    "AssignmentID": f"A{100 + i}",
    "CourseID": random.choice(courses)["CourseID"],
    "Title": fake.sentence(nb_words=4),
    "DueDate": fake.date_between(start_date='-3m', end_date='+3m').isoformat()
} for i in range(NUM_ASSIGNMENTS)]

# 4. Enrollments
enrollments = []
enrollment_pairs = set()
while len(enrollments) < NUM_ENROLLMENTS:
    student = random.choice(students)
    course = random.choice(courses)
    key = (student["StudentID"], course["CourseID"])
    if key not in enrollment_pairs:
        enrollment_pairs.add(key)
        enrollments.append({
            "EnrollmentID": f"E{1000 + len(enrollments)}",
            "StudentID": student["StudentID"],
            "CourseID": course["CourseID"]
        })

# 5. Submissions
submissions = [{
    "SubmissionID": f"SUB{1000 + i}",
    "AssignmentID": random.choice(assignments)["AssignmentID"],
    "StudentID": random.choice(students)["StudentID"],
    "SubmissionDate": fake.date_between(start_date='-2m', end_date='today').isoformat(),
    "Grade": random.choice([None] + [round(random.uniform(60, 100), 2) for _ in range(5)])
} for i in range(NUM_SUBMISSIONS)]
```

*[41]:
```
# Dirty students data
for student in students:
```

```
[41]: # Dirty students data
for student in students:
    if random.random() < 0.05:
        student["Email"] = None  # Missing email
    if random.random() < 0.05:
        student["Name"] = student["Name"].strip().title() + " "  # Extra space
    if random.random() < 0.03:
        student["EnrollmentDate"] = ""  # Empty date string

# Dirty courses data
for course in courses:
    if random.random() < 0.05:
        course["Instructor"] = fake.name().lower()  # lowercase name
    if random.random() < 0.03:
        course["Semester"] = random.choice(["Fall2024", "Spring 2025", "SUMMER 2025"])  # inconsistent format

# Dirty assignments data
for assignment in assignments:
    if random.random() < 0.05:
        assignment["Title"] = assignment["Title"].replace(".", "")  # remove punctuation
    if random.random() < 0.03:
        assignment["DueDate"] = None  # missing due date

# Dirty enrollments data
for enrollment in enrollments:
    if random.random() < 0.03:
        enrollment["CourseID"] = enrollment["CourseID"].lower()  # lowercased ID

# Dirty submissions data
for submission in submissions:
    if random.random() < 0.05:
        submission["Grade"] = None  # Missing grade
    if random.random() < 0.03:
        submission["SubmissionDate"] = submission["SubmissionDate"] + " "  # Trailing space

# Add few duplicates
def add_duplicates(data, percent=0.05):
    num_dupes = int(len(data) * percent)
    duplicates = random.choices(data, k=num_dupes)
    return data + duplicates

# Add duplicates (approx 5% each)
students = add_duplicates(students, percent=0.05)
courses = add_duplicates(courses, percent=0.05)
```


```
# Add few duplicates
def add_duplicates(data, percent=0.05):
    num_dupes = int(len(data) * percent)
    duplicates = random.choices(data, k=num_dupes)
    return data + duplicates

# Add duplicates (approx 5% each)
students = add_duplicates(students, percent=0.05)
courses = add_duplicates(courses, percent=0.05)
assignments = add_duplicates(assignments, percent=0.05)
enrollments = add_duplicates(enrollments, percent=0.05)
submissions = add_duplicates(submissions, percent=0.05)
```


```
[42]: # Save all as CSVs

pd.DataFrame(students).to_csv("students.csv", index=False, quoting=csv.QUOTE_NONE, escapechar='\\')
pd.DataFrame(courses).to_csv("courses.csv", index=False, quoting=csv.QUOTE_NONE, escapechar='\\')
pd.DataFrame(assignments).to_csv("assignments.csv", index=False, quoting=csv.QUOTE_NONE, escapechar='\\')
pd.DataFrame(submissions).to_csv("submissions.csv", index=False, quoting=csv.QUOTE_NONE, escapechar='\\')
pd.DataFrame(enrollments).to_csv("enrollments.csv", index=False, quoting=csv.QUOTE_NONE, escapechar='\\')
```

CSV files (raw dirty data):

jupyter students.csv Last Checkpoint: 5 days ago

File Edit View Settings Help

Delimiter: ,

	StudentID	Name	Email	EnrollmentDate
1	S1000	Allison Hill	ynaldgarcia@example.net	2025-04-25
2	S1001	Leslie Johnson	rsonwilliam@example.org	2024-08-11
3	S1002	Matthew Gardner		2025-01-12
4	S1003	Melissa Peterson	ngallagher@example.org	2023-08-14
5	S1004	Ian Cooper	lindsay78@example.org	2025-05-19
6	S1005	Juan Calderon	barbara10@example.net	2023-11-27
7	S1006	Victoria Wyatt	nesshawn@example.com	2023-10-07
8	S1007	Daniel Adams	mchgeorge@example.net	2023-12-10
9	S1008	James Mayo	llecameron@example.org	2023-11-27
10	S1009	Renee Morales	clarksherri@example.net	2024-02-02
11	S1010	Jonathan Wilkerson	ithomas@example.org	2025-01-01
12	S1011	Deborah Richards	icox@example.net	2024-06-20
13	S1012	Anthony Rodriguez	henderson@example.org	2024-03-27
14	S1013	Zachary Rice	azierdanny@example.net	2025-01-06
15	S1014	Martin Rodriguez	amoore@example.net	2025-04-13
16	S1015	Valerie Gill	rellikenneth@example.net	2024-07-01
17	S1016	Brittany Farmer	eorgetracy@example.org	2025-05-24
18	S1017	William Roman	thumphrey@example.org	2023-08-08
19	S1018	Rose Spence	avidalvarez@example.net	2023-10-09
20	S1019	Shane Henderson	rahcampos@example.net	2023-11-06
21	S1020	Jennifer Powers	samuel87@example.org	2023-12-12
22	S1021	George Chapman	iannasmith@example.net	2024-10-05
23	S1022	Austin Johnson	chad34@example.net	2024-06-14
24	S1023	Carlos Walls	esanchez@example.com	2023-10-28
25	S1024	Matthew Gomez	uowens@example.org	2024-11-07
26	S1025	Maria Brown	lauren13@example.org	2023-10-28
27	S1026	Brenda White	ewright@example.com	2025-02-19
28	S1027	Krista Williams	novaksara@example.org	2023-10-10
29	S1028	John Hancock	artinaaron@example.com	2023-12-25
30	S1029	Rachel Mitchell	smoore@example.org	2023-07-18
31	S1030	Kimberly Adkins	imsthomas@example.net	2023-10-03

jupyter assignments.csv Last Checkpoint: 5 days ago

File Edit View Settings Help

Delimiter: ,

	AssignmentID	CourseID	Title	DueDate
1	A100	C105	Meeting event strong.	2025-06-02
2	A101	C106	Lead certain course.	2025-06-02
3	A102	C109	/ attorney significant take.	2025-06-02
4	A103	C106	vironment able rise study.	2025-06-02
5	A104	C108	nd land machine forward.	2025-06-02
6	A105	C108	ay focus country recently.	2025-06-02
7	A106	C104	Do simply analysis seat.	2025-06-02
8	A107	C109	le society present charge.	2025-06-02
9	A108	C105	Maybe opportunity thank.	2025-06-02
10	A109	C101	Moment lead.	2025-06-02
11	A110	C104	That wide avoid sit.	2025-06-02
12	A111	C100	haps however bag forget.	2025-06-02
13	A112	C109	t phone thought maintain.	2025-06-02
14	A113	C106	Pay doctor.	2025-06-02
15	A114	C105	Section its better.	2025-06-02
16	A115	C108	Kitchen really.	2025-06-02
17	A116	C104	Food argue.	2025-06-02
18	A117	C102	Issue each include radio.	2025-06-02
19	A118	C102	Company participant him.	2025-06-02
20	A119	C104	Tree central leave effect.	2025-06-02
21	A120	C102	Together let.	2025-06-02
22	A121	C109	id generation onto police.	2025-06-02
23	A122	C101	Lead few yourself.	2025-06-02
24	A123	C102	Down occur.	2025-06-02
25	A124	C104	Yeah far within	2025-06-02
26	A125	C100	Ever not rate seat	2025-06-02
27	A126	C109	legy total simply discover.	2025-06-02
28	A127	C102	Pm per question.	2025-06-03
29	A128	C100	Pick tough position.	2025-06-02
30	A129	C100	Almost half capital travel.	2025-06-02
31	A130	C107	Compare car much will.	2025-06-02

jupyter courses.csv Last Checkpoint: 5 days ago

File Edit View Settings Help

Delimiter: ,

	CourseID	CourseName	Instructor	Semester
1	C100	even-keeled contingency	Jessica Olsen DVM	Summer 2025
2	C101	d grid-enabled installation	Michael Hodge	Fall 2024
3	C102	aged incremental solution	Matthew Morales DDS	Spring 2025
4	C103	Automated logistical array	Frank Cordova	Fall 2024
5	C104	hitected multimedia ability	Kimberly Gutierrez	Fall 2024
6	C105	omic multi-state hierarchy	Kathleen Ramos	Spring 2025
7	C106	d demand-driven flexibility	Tanya Kim	Summer 2025
8	C107	Total mobile capability	Mark Lloyd	Fall 2024
9	C108	uffered cohesive function	Kristen Davis	Summer 2025
10	C109	able maximized capability	Brittany Cantu	Fall 2024

jupyter enrollments.csv Last Checkpoint: 5 days ago

File Edit View Settings Help

Delimiter: ,

	EnrollmentID	StudentID	CourseID
1	E1000	S1049	C109
2	E1001	S1041	C102
3	E1002	S1004	C100
4	E1003	S1005	c109
5	E1004	S1003	C104
6	E1005	S1091	C107
7	E1006	S1089	C106
8	E1007	S1006	C103
9	E1008	S1036	C106
10	E1009	S1040	C109
11	E1010	S1052	C100
12	E1011	S1044	C105
13	E1012	S1081	C105
14	E1013	S1026	C102
15	E1014	S1078	C106
16	E1015	S1047	C109
17	E1016	S1044	C108
18	E1017	S1035	C105
19	E1018	S1048	c106
20	E1019	S1001	C108
21	E1020	S1090	C104
22	E1021	S1057	C107
23	E1022	S1003	C102
24	E1023	S1083	C107
25	E1024	S1060	C109
26	E1025	S1075	C105
27	E1026	S1042	C100
28	E1027	S1053	C101
29	E1028	S1097	C100
30	E1029	S1097	C105
31	E1030	S1011	C102

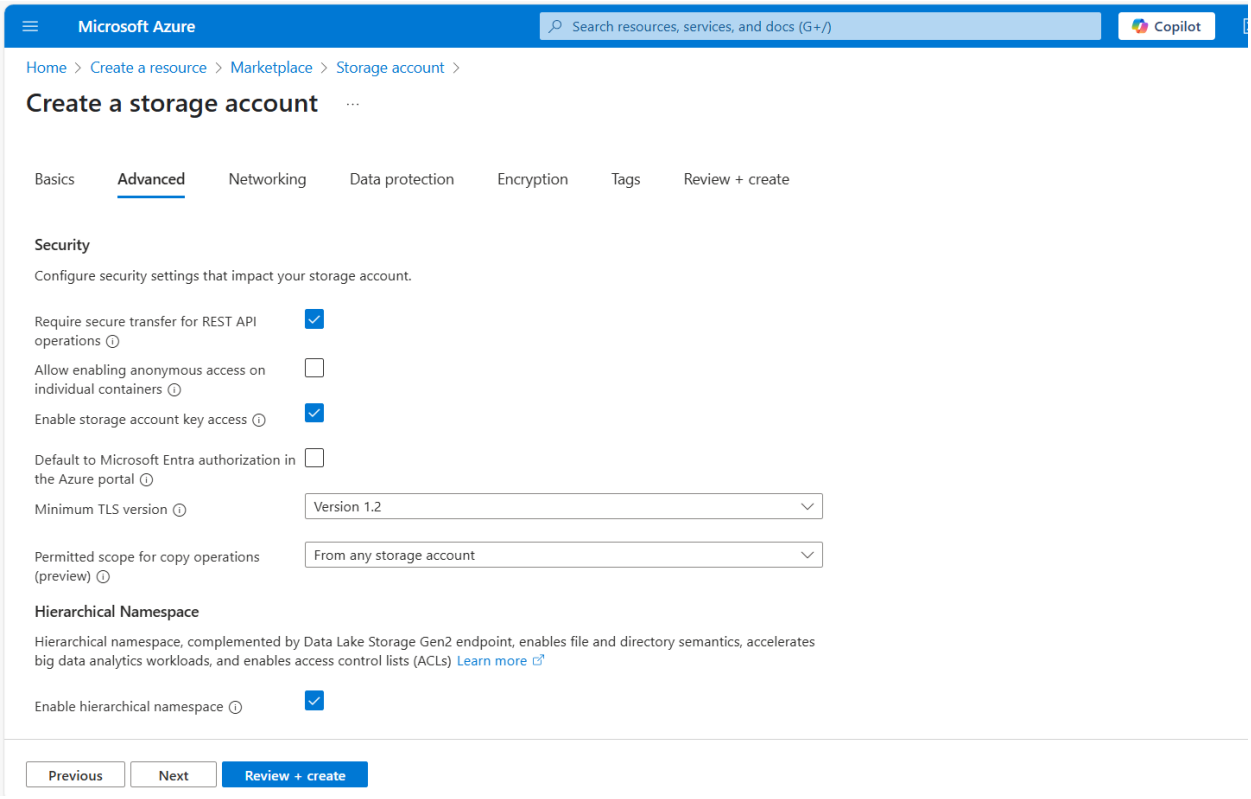
jupyter submissions.csv Last Checkpoint: 5 days ago

File Edit View Settings Help

Delimiter: ,

	SubmissionID	AssignmentID	StudentID	SubmissionDate	Grade
1	SUB1000	A112	S1052	2025-06-02	90.36
2	SUB1001	A101	S1056	2025-06-02	
3	SUB1002	A148	S1021	2025-06-02	69.87
4	SUB1003	A111	S1075	2025-06-02	67.99
5	SUB1004	A100	S1057	2025-06-02	
6	SUB1005	A148	S1000	2025-06-02	74.39
7	SUB1006	A134	S1048	2025-06-02	85.92
8	SUB1007	A110	S1089	2025-06-02	
9	SUB1008	A141	S1016	2025-06-02	61.64
10	SUB1009	A143	S1000	2025-06-02	
11	SUB1010	A113	S1034	2025-06-02	
12	SUB1011	A109	S1042	2025-06-02	
13	SUB1012	A121	S1073	2025-06-02	98.06
14	SUB1013	A124	S1083	2025-06-02	67.96
15	SUB1014	A139	S1080	2025-06-02	63.84
16	SUB1015	A146	S1036	2025-06-02	75.8
17	SUB1016	A144	S1048	2025-06-02	94.35
18	SUB1017	A133	S1089	2025-06-02	63.1
19	SUB1018	A115	S1000	2025-06-02	76.26
20	SUB1019	A110	S1014	2025-06-02	97.34
21	SUB1020	A106	S1008	2025-06-02	64.61
22	SUB1021	A147	S1005	2025-06-02	92.17
23	SUB1022	A129	S1048	2025-06-02	94.53
24	SUB1023	A145	S1022	2025-06-02	89.18
25	SUB1024	A149	S1016	2025-06-02	
26	SUB1025	A134	S1073	2025-06-02	82.23
27	SUB1026	A109	S1006	2025-06-02	84.38
28	SUB1027	A103	S1033	2025-06-02	
29	SUB1028	A146	S1031	2025-06-02	92.68
30	SUB1029	A130	S1026	2025-06-02	66.02
31	SUB1030	A148	S1025	2025-06-02	96.08

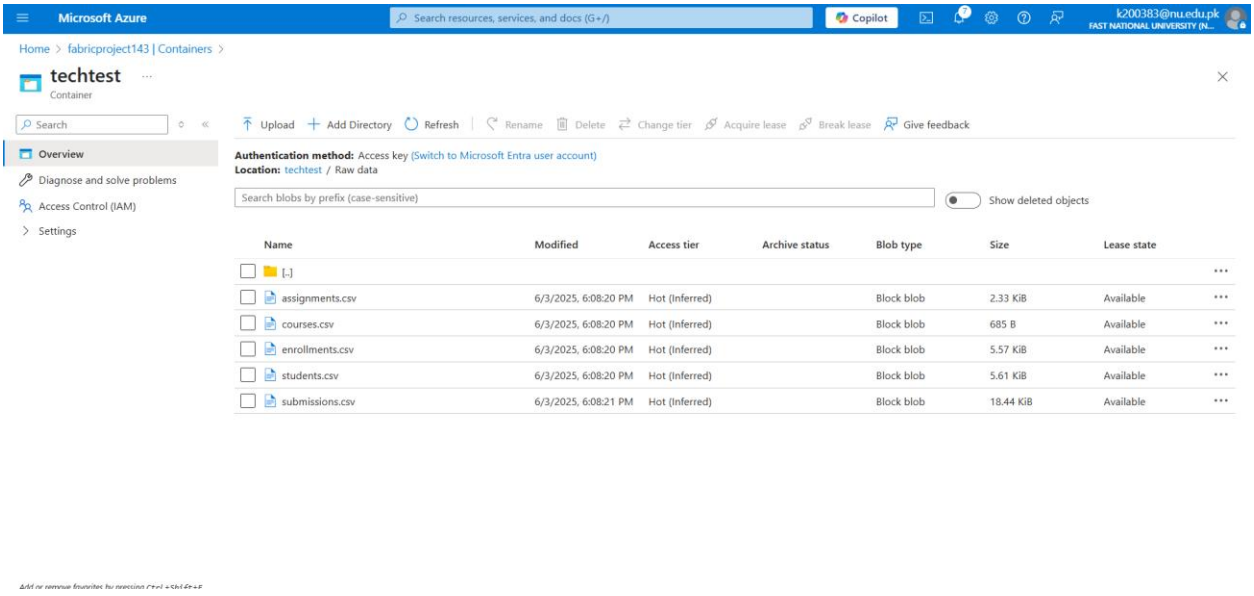
Create storage account in Azure:



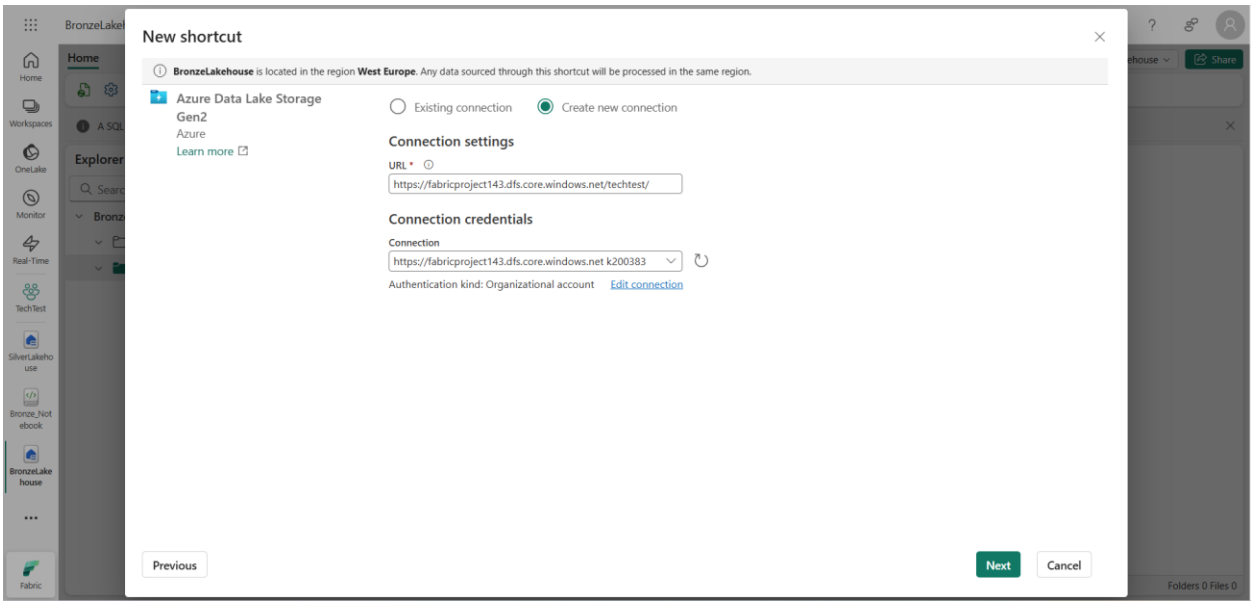
(ticked hierarchal namespace to make it ADLS and not blob storage)

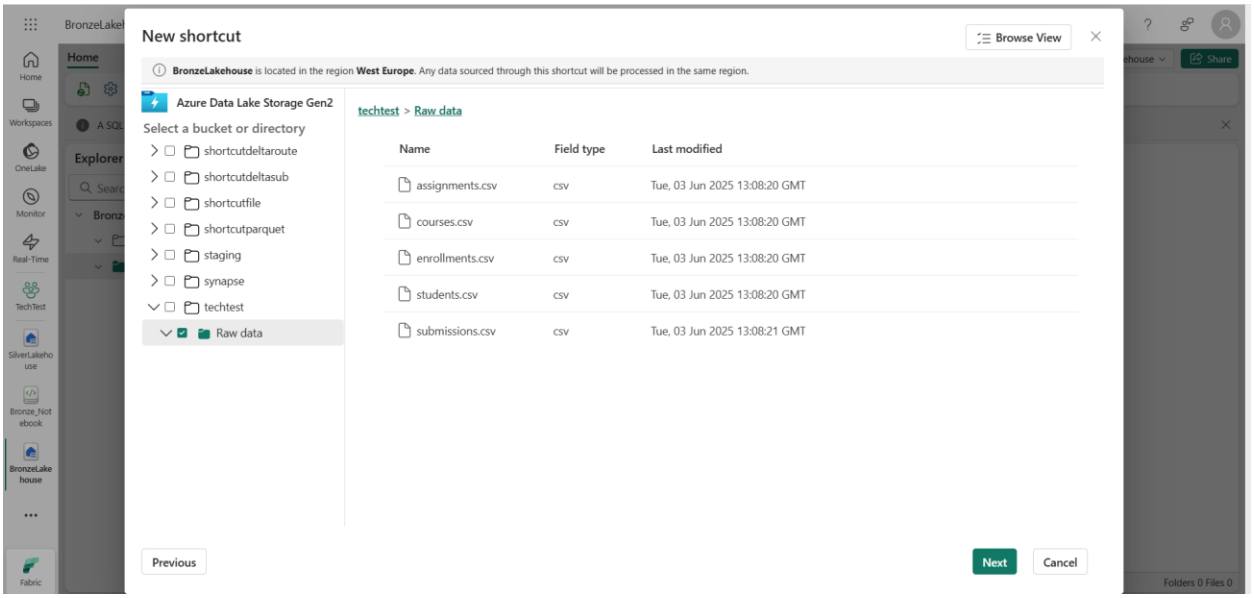
Here, I created the resource group (by clicking on storage account), created ADLS and then an associated container.

Raw data in ADLS (CSV files):

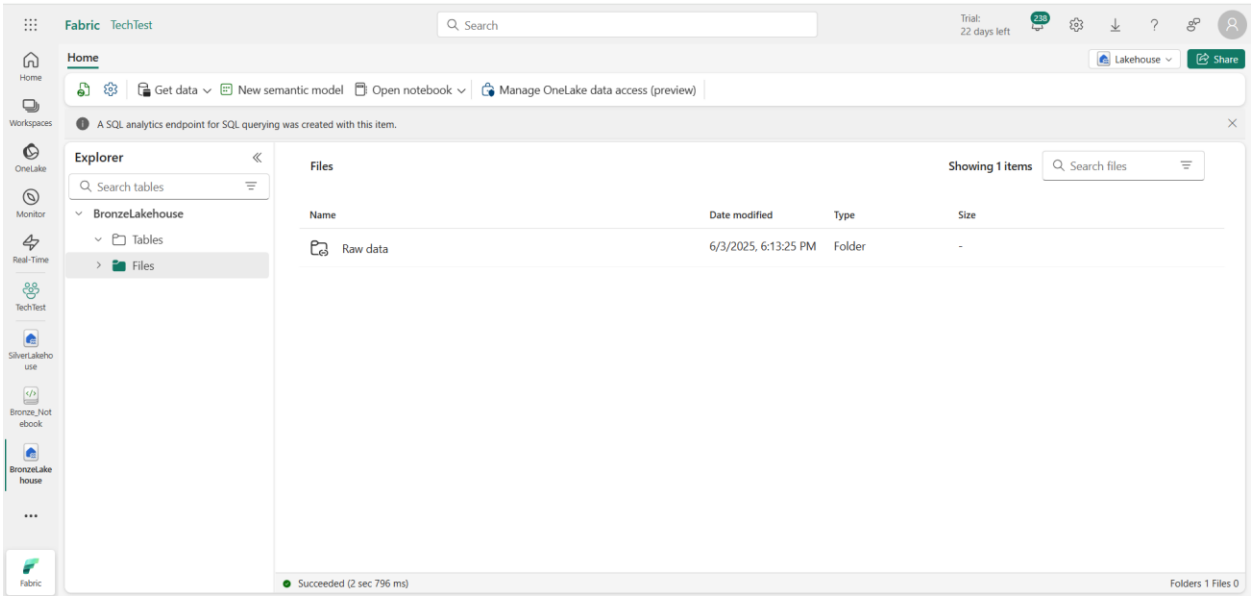


Create shortcut in fabric lakehouse (bronze) from ADLS:

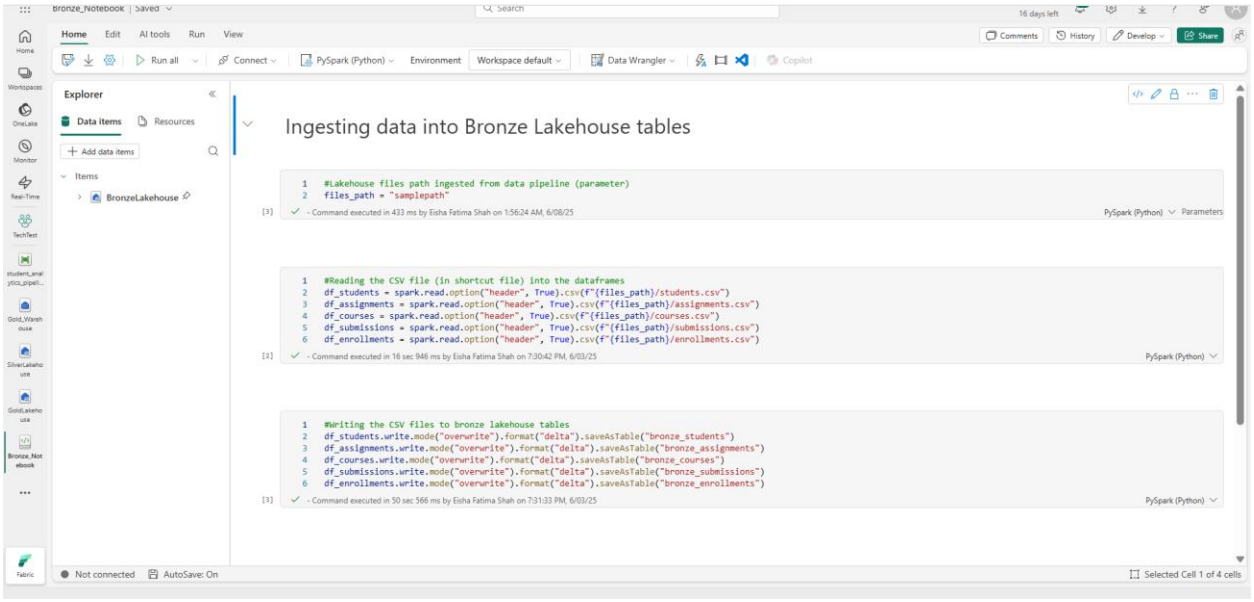




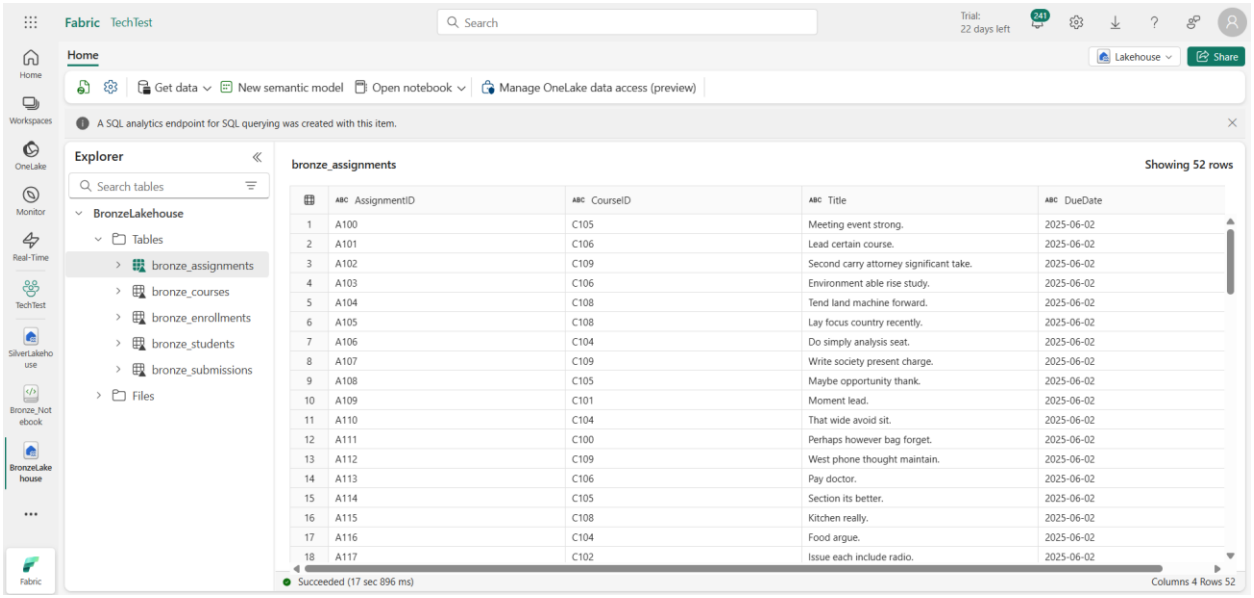
Bronze lakehouse preview:



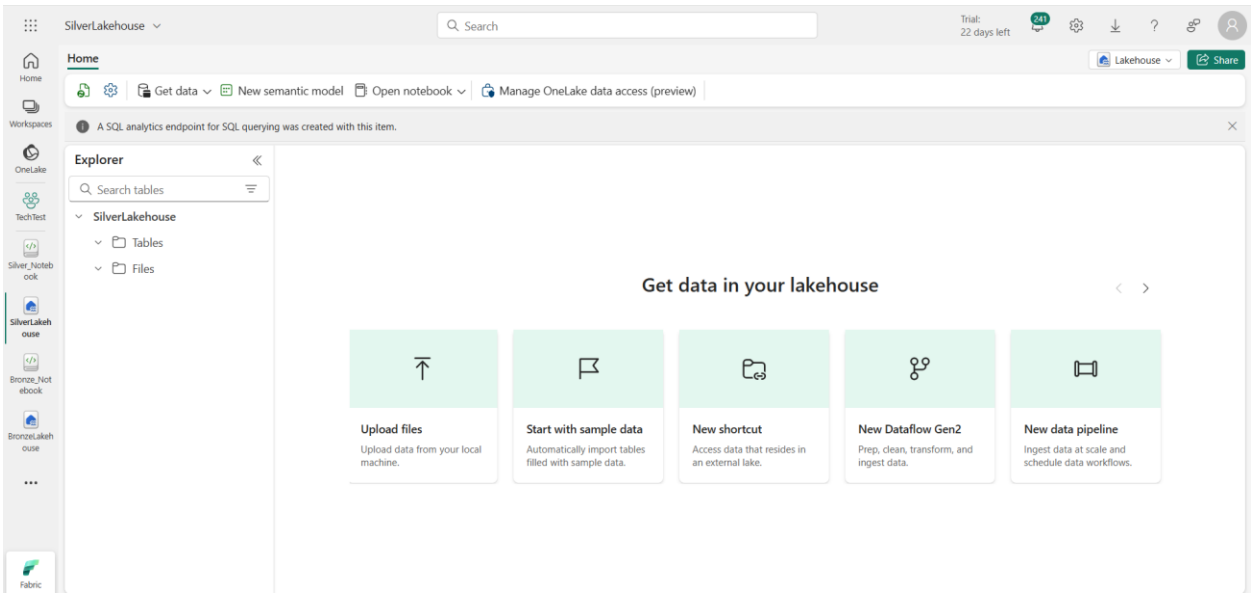
Ingesting data into bronze lakehouse:



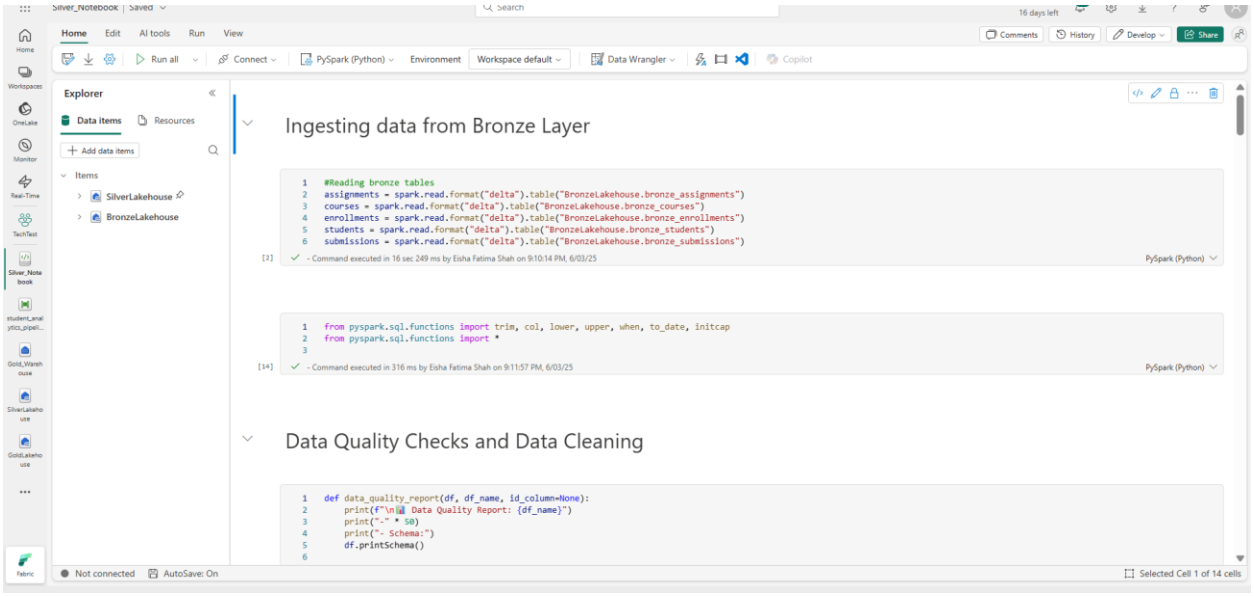
Successfully created bronze tables:

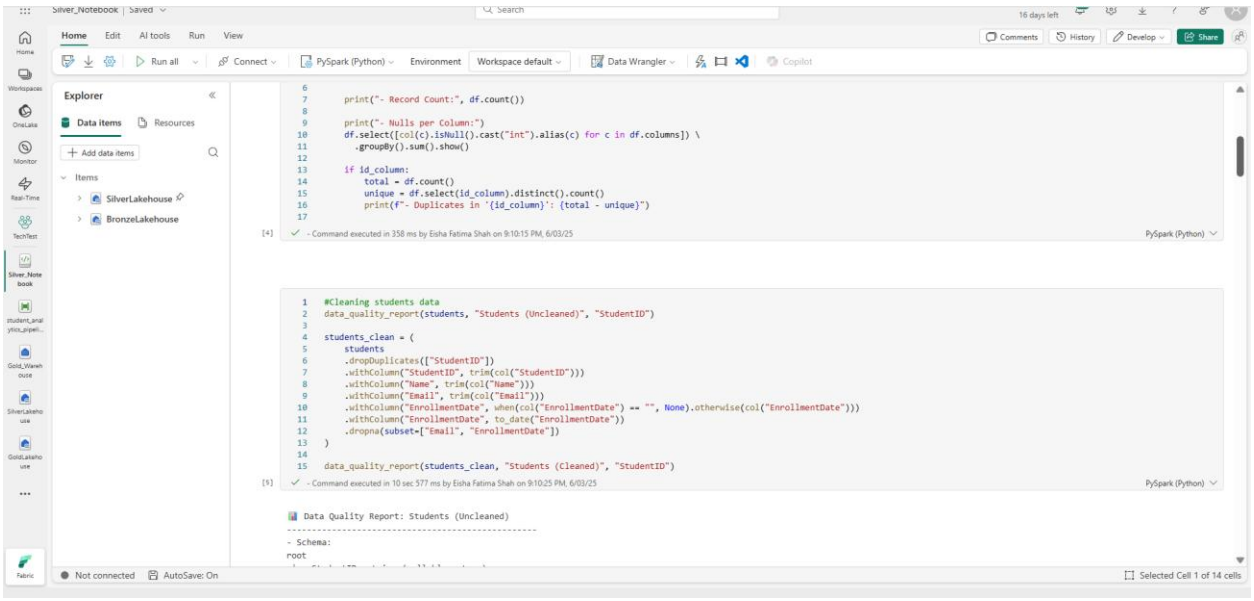


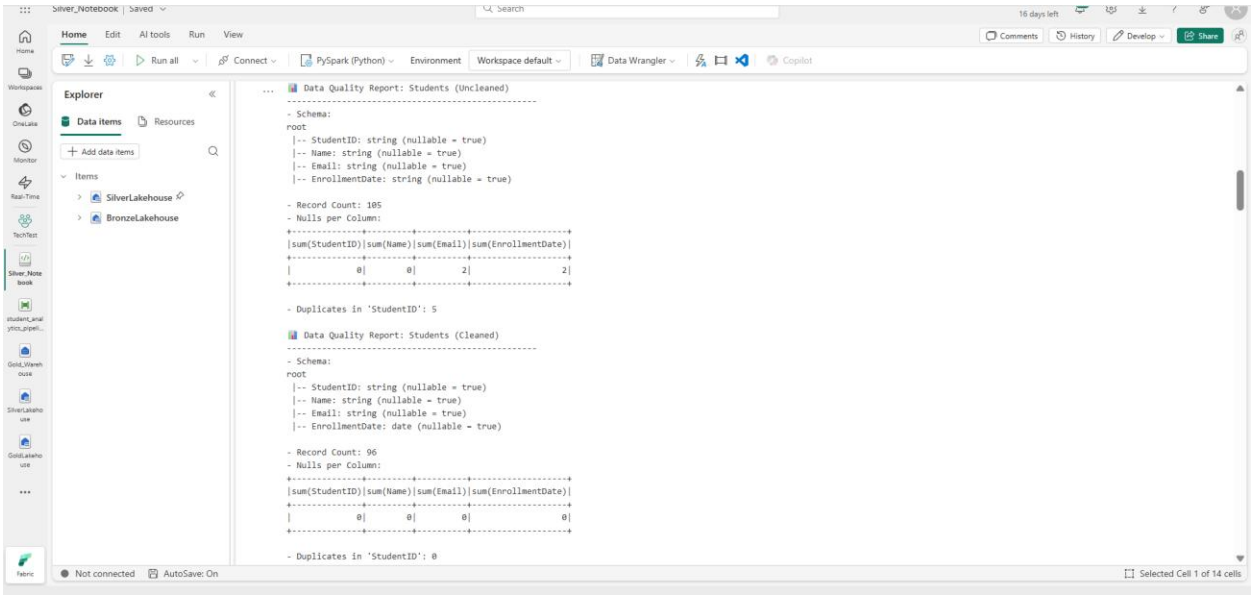
Preview of silver lakehouse:

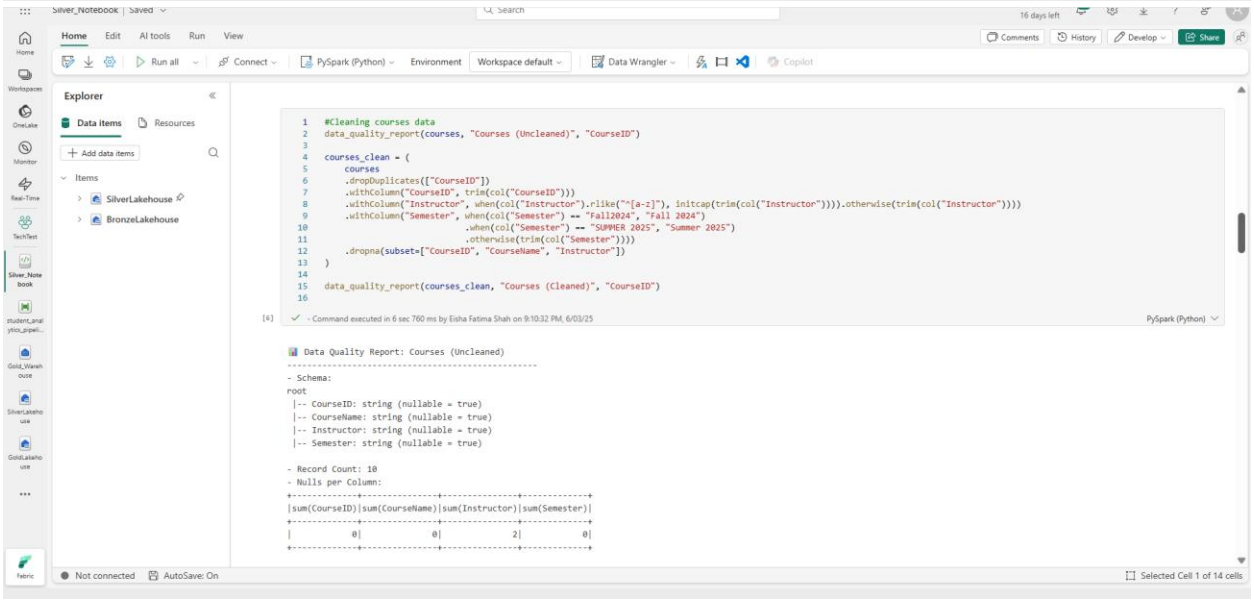


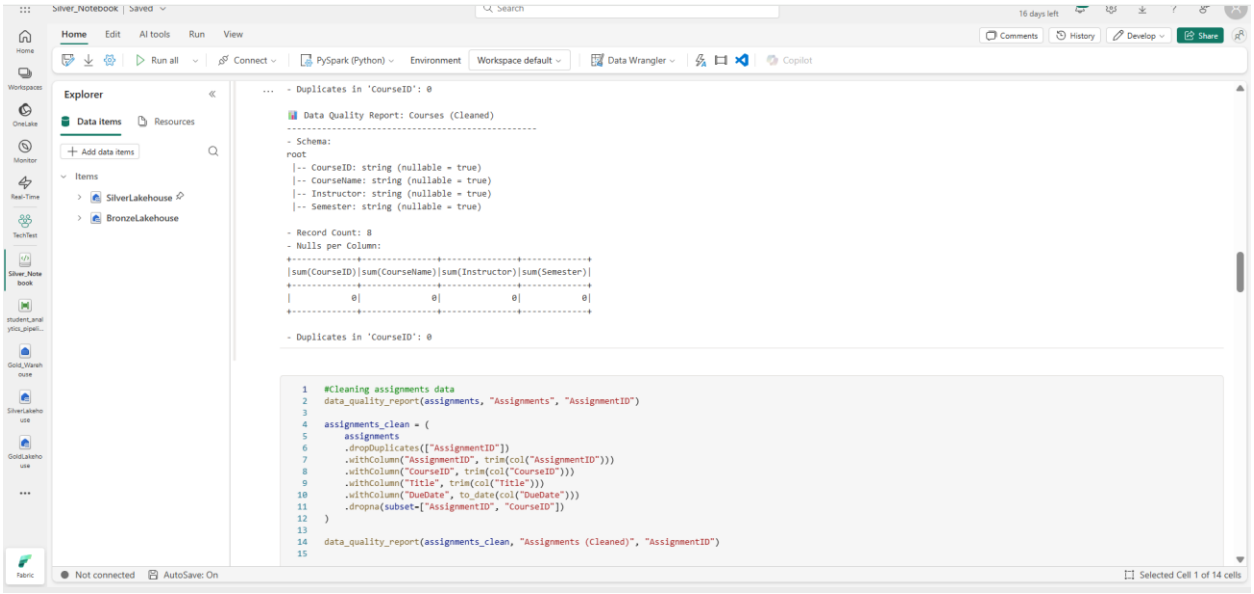
Coding the silver layer in notebook:

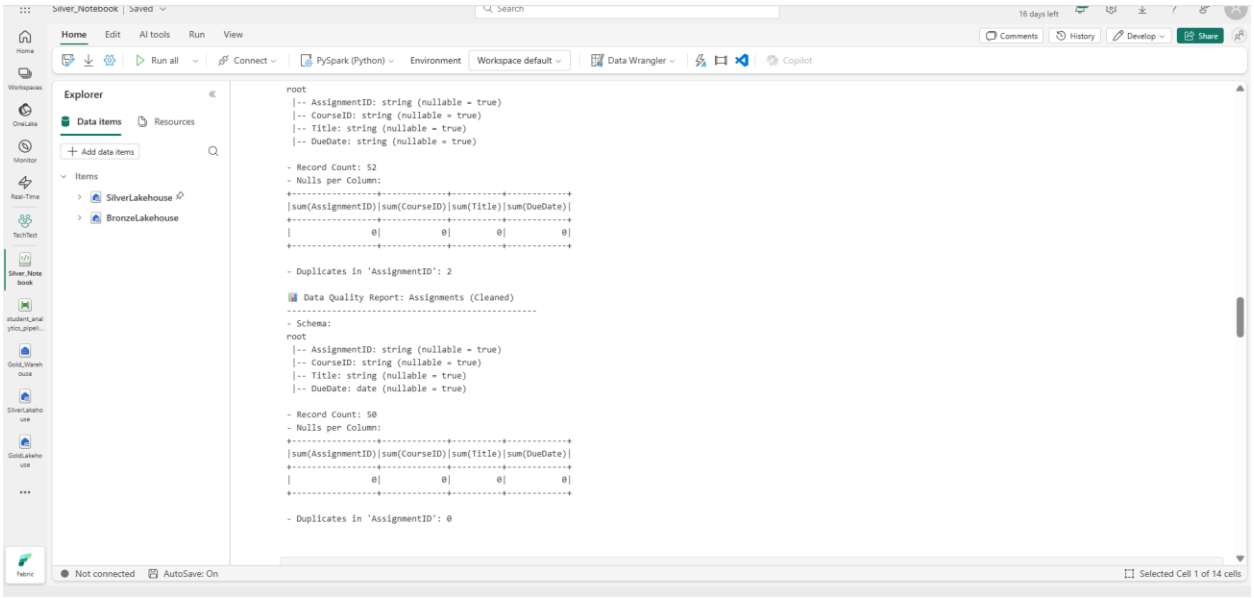


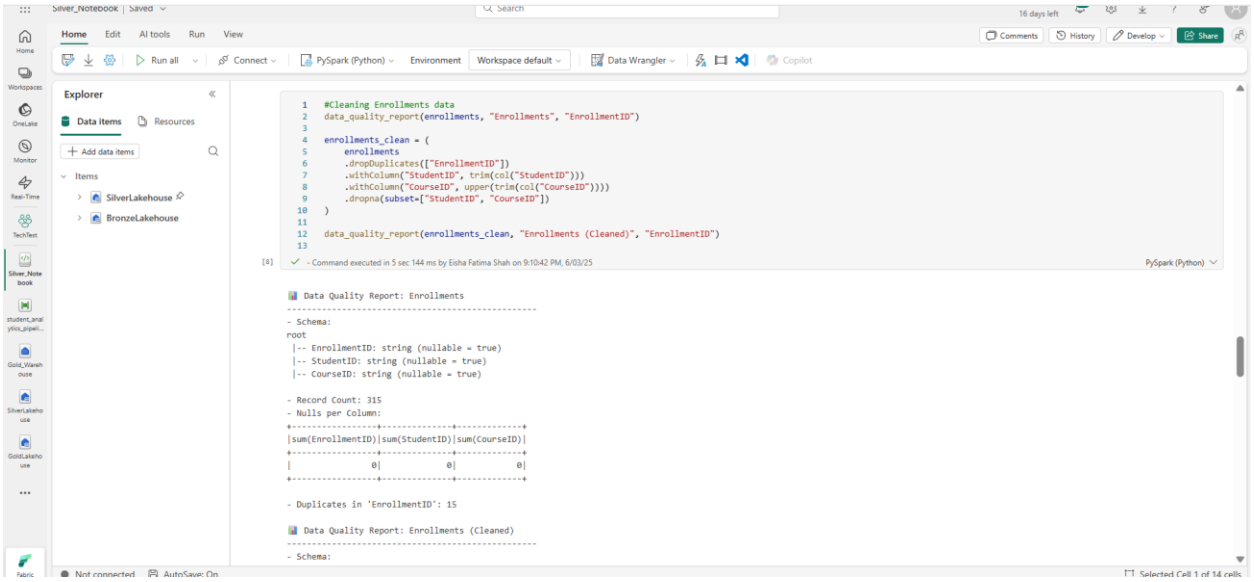


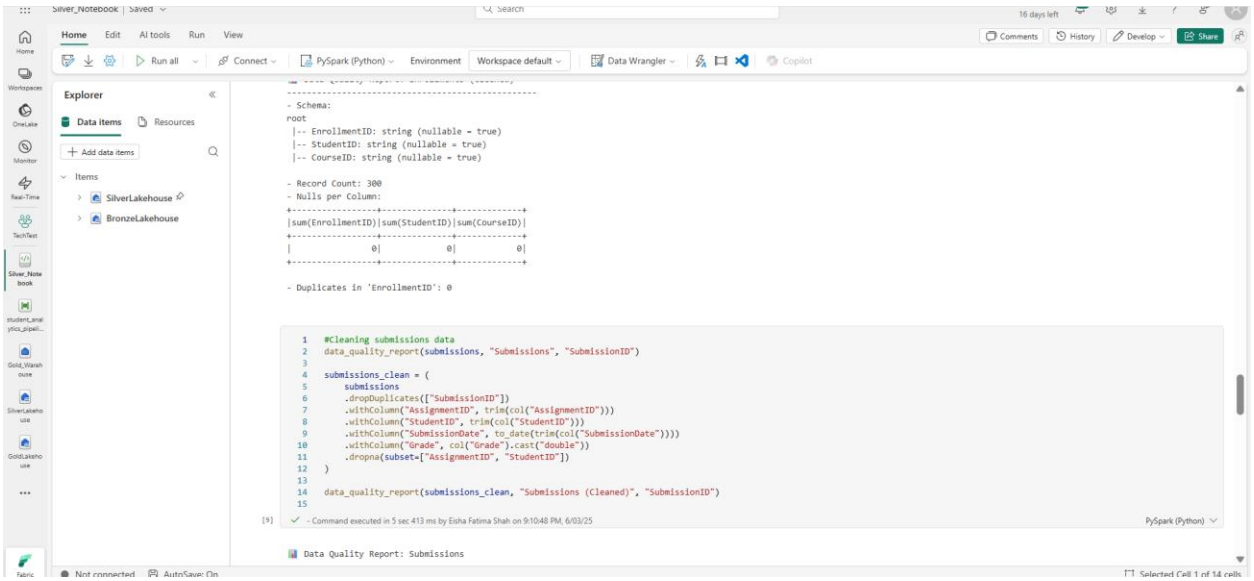

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
Data Quality Report: Students (Uncleaned)
- Schema:
root
|-- StudentID: string (nullable = true)
|-- Name: string (nullable = true)
|-- Email: string (nullable = true)
|-- EnrollmentDate: string (nullable = true)
- Record Count: 105
- Nulls per Column:
+--------------+---------+----------+------------------+
|sum(StudentID)|sum(Name)|sum(Email)|sum(EnrollmentDate)|
+--------------+---------+----------+------------------+
| 0| 0| 2| 2|
+--------------+---------+----------+------------------+
- Duplicates in 'StudentID': 5
Data Quality Report: Students (Cleaned)
- Schema:
root
|-- StudentID: string (nullable = true)
|-- Name: string (nullable = true)
|-- Email: string (nullable = true)
|-- EnrollmentDate: date (nullable = true)
- Record Count: 96
- Nulls per Column:
+--------------+---------+----------+------------------+
|sum(StudentID)|sum(Name)|sum(Email)|sum(EnrollmentDate)|
+--------------+---------+----------+------------------+
| 0| 0| 0| 0|
+--------------+---------+----------+------------------+
- Duplicates in 'StudentID': 0
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

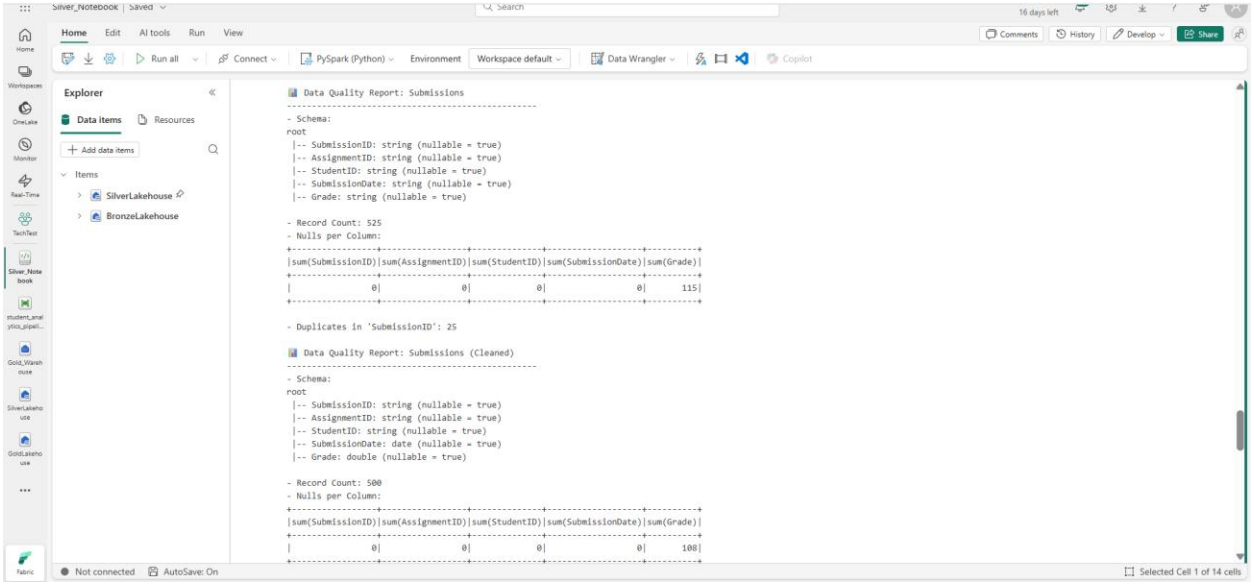

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
#Cleaning courses data
data_quality_report(courses, "Courses (Uncleaned)", "CourseID")
courses_clean = (
courses
.dropDuplicates(["CourseID"])
.withColumn("CourseID", trim(col("CourseID")))
.withColumn("Instructor", when(col("Instructor").rlike("^[a-z]"), initcap(trim(col("Instructor")))).otherwise(trim(col("Instructor"))))
.withColumn("Semester", when(col("Semester") == "Fall2024", "Fall 2024")
.when(col("Semester") == "SUMMER 2025", "Summer 2025")
.otherwise(trim(col("Semester"))))
.dropna(subset=["CourseID", "CourseName", "Instructor"])
)
data_quality_report(courses_clean, "Courses (Cleaned)", "CourseID")
[6] - Command executed in 6 sec 760 ms by Eisha Fatima Shah on 9:10:32 PM, 6/03/25
PySpark (Python)
Data Quality Report: Courses (Uncleaned)
- Schema:
root
|-- CourseID: string (nullable = true)
|-- CourseName: string (nullable = true)
|-- Instructor: string (nullable = true)
|-- Semester: string (nullable = true)
- Record Count: 10
- Nulls per Column:
+--------------+----------------+--------------+------------+
|sum(CourseID)|sum(CourseName)|sum(Instructor)|sum(Semester)|
+--------------+----------------+--------------+------------+
| 0| 0| 2| 0|
+--------------+----------------+--------------+------------+
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

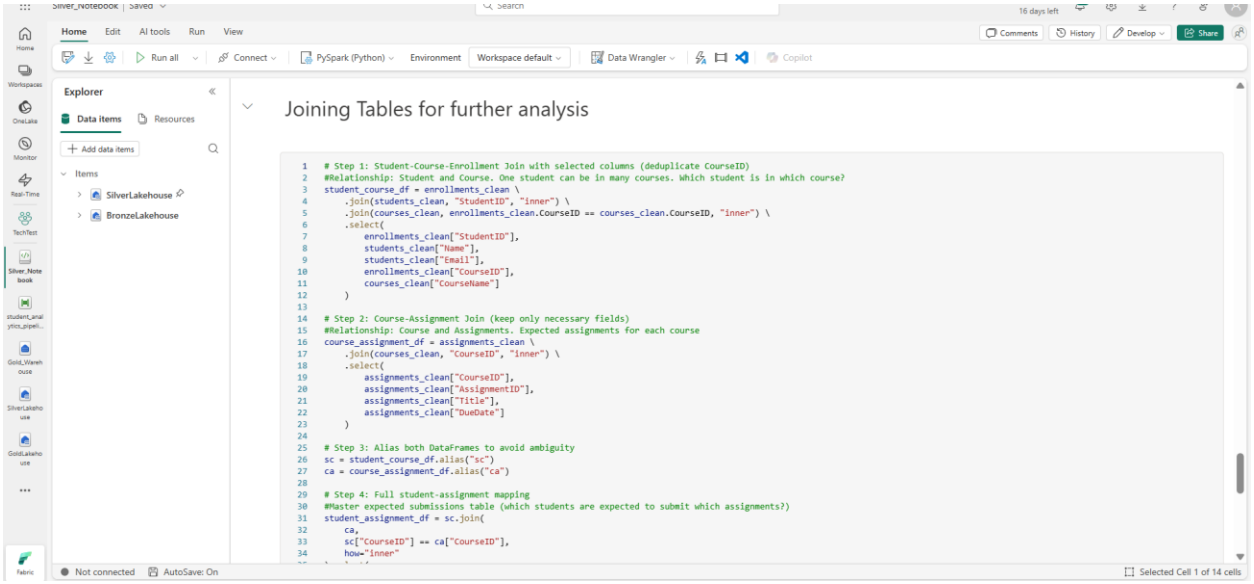

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
- Duplicates in 'CourseID': 0
Data Quality Report: Courses (Cleaned)
- Schema:
root
|-- CourseID: string (nullable = true)
|-- CourseName: string (nullable = true)
|-- Instructor: string (nullable = true)
|-- Semester: string (nullable = true)
- Record Count: 8
- Nulls per Column:
|sum(CourseID)|sum(CourseName)|sum(Instructor)|sum(Semester)|
| 0| 0| 0| 0|
- Duplicates in 'CourseID': 0
#Cleaning assignments data
data_quality_report(assignments, "Assignments", "AssignmentID")
assignments_clean = (
assignments
.dropDuplicates(["AssignmentID"])
.withColumn("AssignmentID", trim(col("AssignmentID")))
.withColumn("CourseID", trim(col("CourseID")))
.withColumn("Title", trim(col("Title")))
.withColumn("DueDate", to_date(col("DueDate")))
.dropna(subset=["AssignmentID", "CourseID"])
)
data_quality_report(assignments_clean, "Assignments (Cleaned)", "AssignmentID")
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

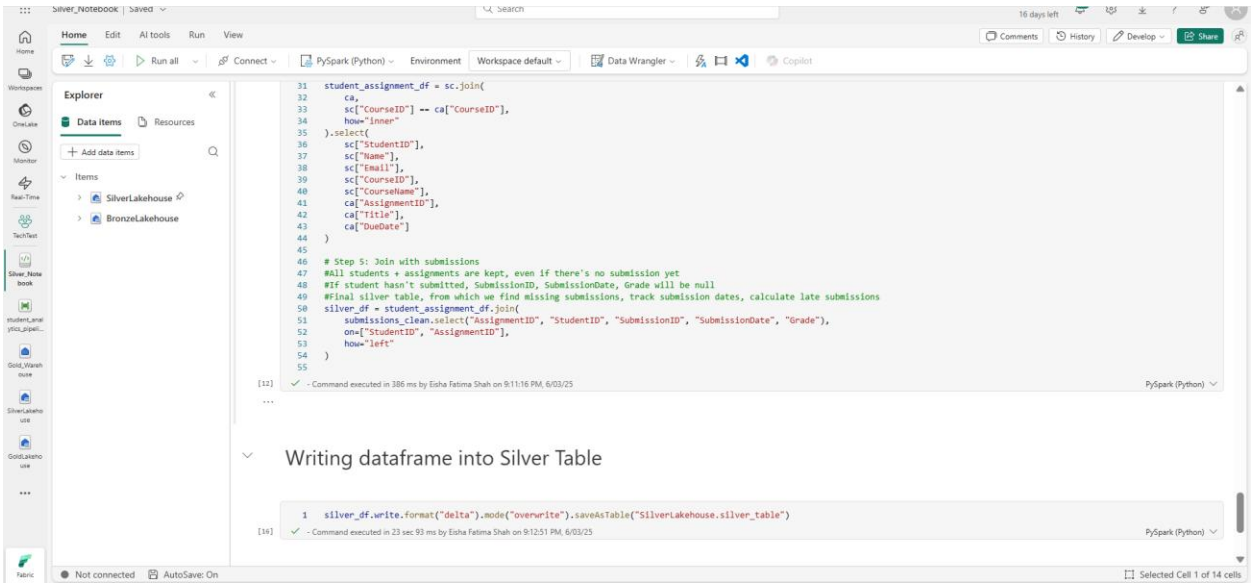

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
root
|-- AssignmentID: string (nullable = true)
|-- CourseID: string (nullable = true)
|-- Title: string (nullable = true)
|-- DueDate: string (nullable = true)
- Record Count: 52
- Nulls per Column:
|sum(AssignmentID)|sum(CourseID)|sum(Title)|sum(DueDate)|
| 0| 0| 0| 0|
- Duplicates in 'AssignmentID': 2
Data Quality Report: Assignments (Cleaned)
- Schema:
root
|-- AssignmentID: string (nullable = true)
|-- CourseID: string (nullable = true)
|-- Title: string (nullable = true)
|-- DueDate: date (nullable = true)
- Record Count: 50
- Nulls per Column:
|sum(AssignmentID)|sum(CourseID)|sum(Title)|sum(DueDate)|
| 0| 0| 0| 0|
- Duplicates in 'AssignmentID': 0
Not connected
AutoSave: On
Selected Cell 1 of 14 cells


Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
#Cleaning Enrollments data
data_quality_report(enrollments, "Enrollments", "EnrollmentID")
enrollments_clean = (
enrollments
.dropDuplicates(["EnrollmentID"])
.withColumn("StudentID", trim(col("StudentID")))
.withColumn("CourseID", upper(trim(col("CourseID"))))
.dropna(subset=["StudentID", "CourseID"])
)
data_quality_report(enrollments_clean, "Enrollments (Cleaned)", "EnrollmentID")
[8]
- Command executed in 5 sec 144 ms by Eisha Fatima Shah on 9:10:42 PM, 6/03/25
PySpark (Python)
Data Quality Report: Enrollments
- Schema:
root
|-- EnrollmentID: string (nullable = true)
|-- StudentID: string (nullable = true)
|-- CourseID: string (nullable = true)
- Record Count: 315
- Nulls per Column:
|sum(EnrollmentID)|sum(StudentID)|sum(CourseID)|
| 0| 0| 0|
- Duplicates in 'EnrollmentID': 15
Data Quality Report: Enrollments (Cleaned)
- Schema:
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

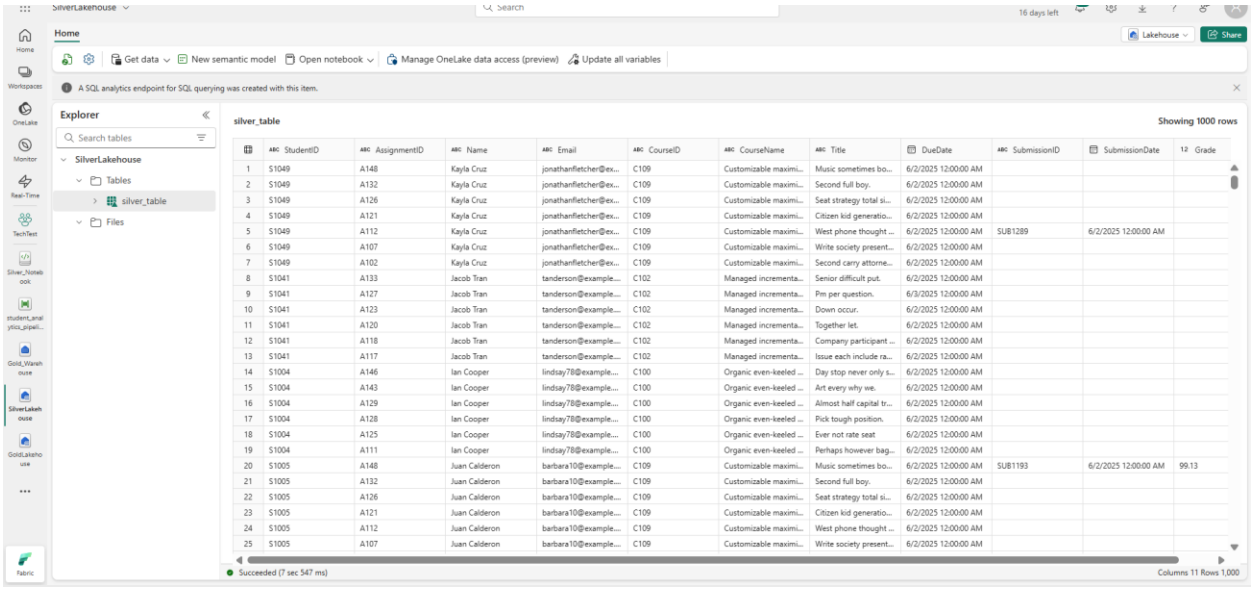

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
- Schema:
root
|-- EnrollmentID: string (nullable = true)
|-- StudentID: string (nullable = true)
|-- CourseID: string (nullable = true)
- Record Count: 300
- Nulls per Column:
|sum(EnrollmentID)|sum(StudentID)|sum(CourseID)|
| 0| 0| 0|
- Duplicates in 'EnrollmentID': 0
#Cleaning submissions data
data_quality_report(submissions, "Submissions", "SubmissionID")
submissions_clean = (
submissions
.dropDuplicates(["SubmissionID"])
.withColumn("AssignmentID", trim(col("AssignmentID")))
.withColumn("StudentID", trim(col("StudentID")))
.withColumn("SubmissionDate", to_date(trim(col("SubmissionDate"))))
.withColumn("Grade", col("Grade").cast("double"))
.dropna(subset=["AssignmentID", "StudentID"])
)
data_quality_report(submissions_clean, "Submissions (Cleaned)", "SubmissionID")
[9]
- Command executed in 5 sec 413 ms by Eisha Fatima Shah on 9:10:48 PM, 6/03/25
PySpark (Python)
Data Quality Report: Submissions
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

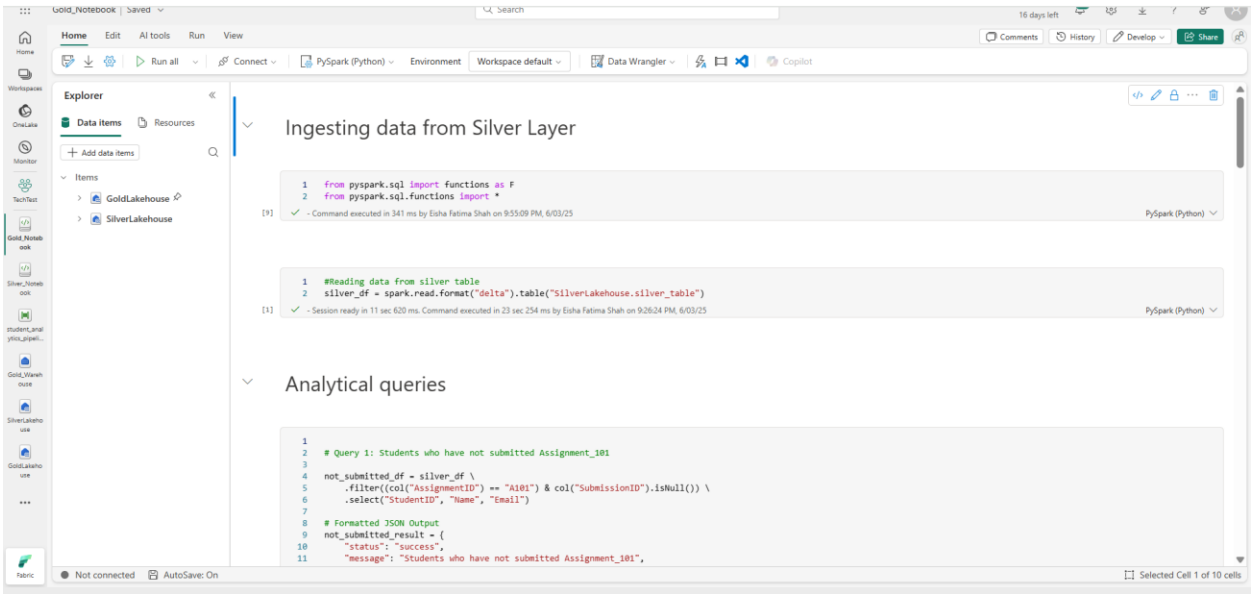

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
📊 Data Quality Report: Submissions
- Schema:
root
|-- SubmissionID: string (nullable = true)
|-- AssignmentID: string (nullable = true)
|-- StudentID: string (nullable = true)
|-- SubmissionDate: string (nullable = true)
|-- Grade: string (nullable = true)
- Record Count: 525
- Nulls per Column:
|sum(SubmissionID)|sum(AssignmentID)|sum(StudentID)|sum(SubmissionDate)|sum(Grade)|
| 0| 0| 0| 0| 115|
- Duplicates in 'SubmissionID': 25
📊 Data Quality Report: Submissions (Cleaned)
- Schema:
root
|-- SubmissionID: string (nullable = true)
|-- AssignmentID: string (nullable = true)
|-- StudentID: string (nullable = true)
|-- SubmissionDate: date (nullable = true)
|-- Grade: double (nullable = true)
- Record Count: 500
- Nulls per Column:
|sum(SubmissionID)|sum(AssignmentID)|sum(StudentID)|sum(SubmissionDate)|sum(Grade)|
| 0| 0| 0| 0| 108|
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

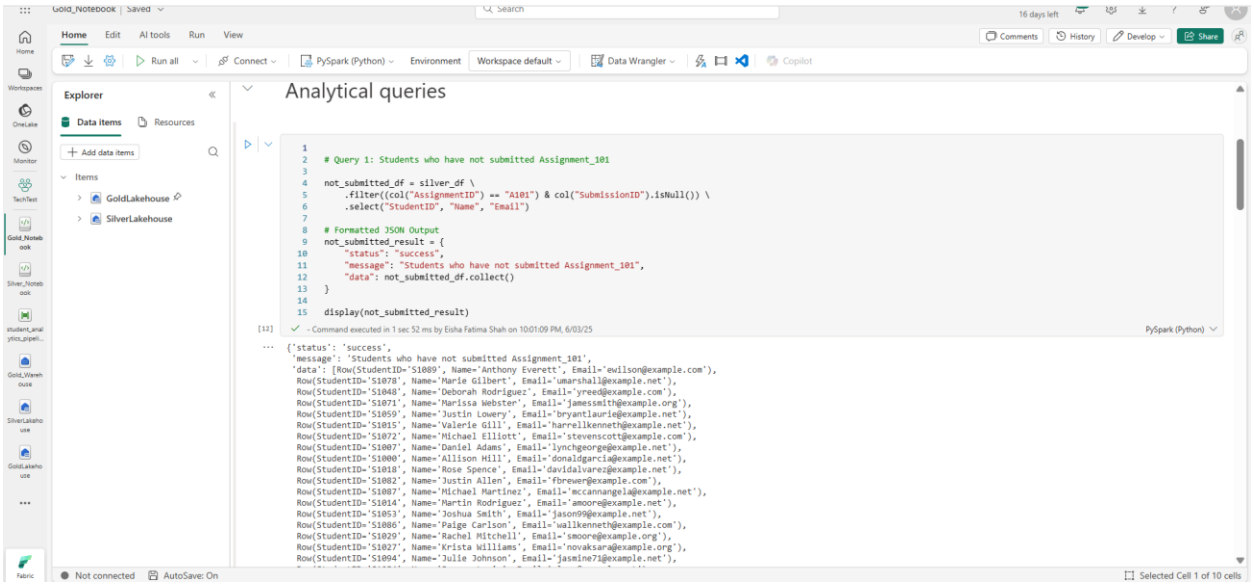

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
Joining Tables for further analysis
1 # Step 1: Student-Course-Enrollment Join with selected columns (deduplicate CourseID)
2 #Relationship: Student and Course. One student can be in many courses. Which student is in which course?
3 student_course_df = enrollments_clean \
4 .join(students_clean, "StudentID", "inner") \
5 .join(courses_clean, enrollments_clean.CourseID == courses_clean.CourseID, "inner") \
6 .select(
7 enrollments_clean["StudentID"],
8 students_clean["Name"],
9 students_clean["Email"],
10 enrollments_clean["CourseID"],
11 courses_clean["CourseName"]
12)
13
14 # Step 2: Course-Assignment Join (keep only necessary fields)
15 #Relationship: Course and Assignments. Expected assignments for each course
16 course_assignment_df = assignments_clean \
17 .join(courses_clean, "CourseID", "inner") \
18 .select(
19 assignments_clean["CourseID"],
20 assignments_clean["AssignmentID"],
21 assignments_clean["Title"],
22 assignments_clean["DueDate"]
23)
24
25 # Step 3: Alias both DataFrames to avoid ambiguity
26 sc = student_course_df.alias("sc")
27 ca = course_assignment_df.alias("ca")
28
29 # Step 4: Full student-assignment mapping
30 #Master expected submissions table (which students are expected to submit which assignments?)
31 student_assignment_df = sc.join(
32 ca,
33 sc["CourseID"] == ca["CourseID"],
34 how="inner"
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

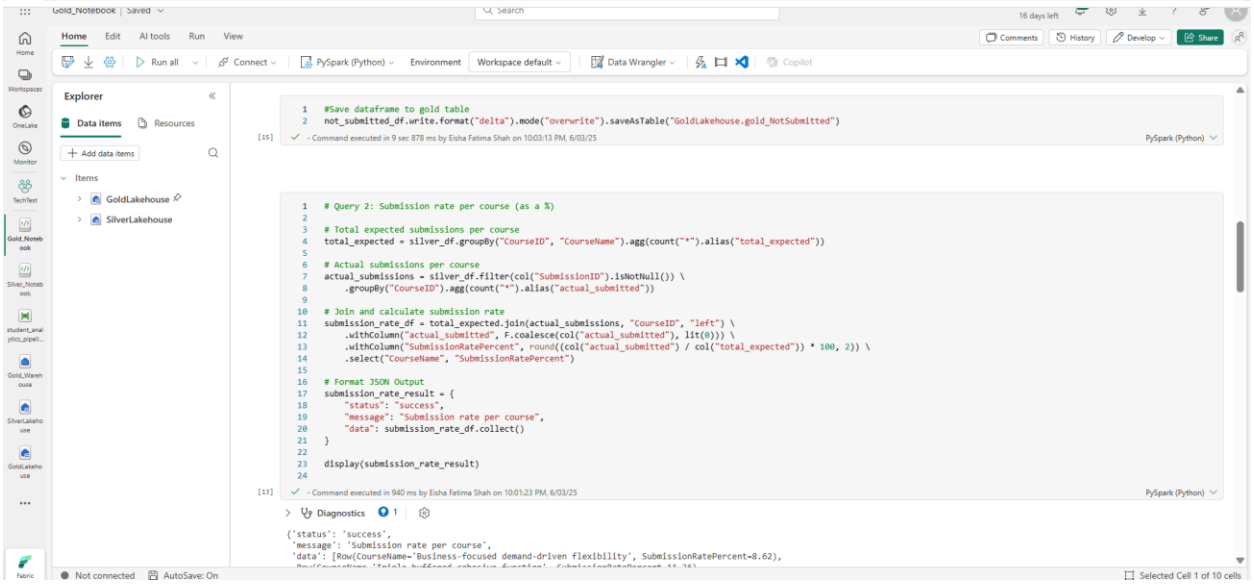

Silver_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
SilverLakehouse
BronzeLakehouse
31 student_assignment_df = sc.join(
32 ca,
33 sc["CourseID"] == ca["CourseID"],
34 how="inner"
35).select(
36 sc["StudentID"],
37 sc["Name"],
38 sc["Email"],
39 sc["CourseID"],
40 sc["CourseName"],
41 ca["AssignmentID"],
42 ca["Title"],
43 ca["DueDate"]
44)
45
46 # Step 5: Join with submissions
47 #All students + assignments are kept, even if there's no submission yet
48 #If student hasn't submitted, SubmissionID, SubmissionDate, Grade will be null
49 #Final silver table, from which we find missing submissions, track submission dates, calculate late submissions
50 silver_df = student_assignment_df.join(
51 submissions_clean.select("AssignmentID", "StudentID", "SubmissionID", "SubmissionDate", "Grade"),
52 on=["StudentID", "AssignmentID"],
53 how="left"
54)
55
[12] - Command executed in 386 ms by Eisha Fatima Shah on 9:11:16 PM, 6/03/25
PySpark (Python)
Writing dataframe into Silver Table
1 silver_df.write.format("delta").mode("overwrite").saveAsTable("SilverLakehouse.silver_table")
[16] - Command executed in 23 sec 93 ms by Eisha Fatima Shah on 9:12:51 PM, 6/03/25
PySpark (Python)
Not connected
AutoSave: On
Selected Cell 1 of 14 cells

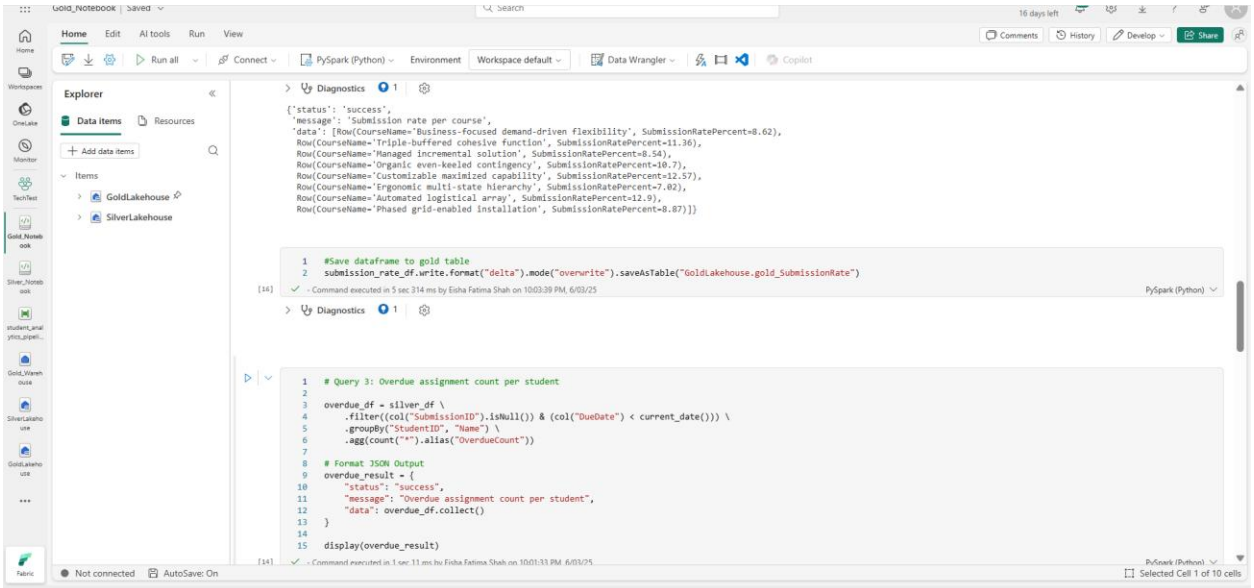
Silver table in silver lakehouse:

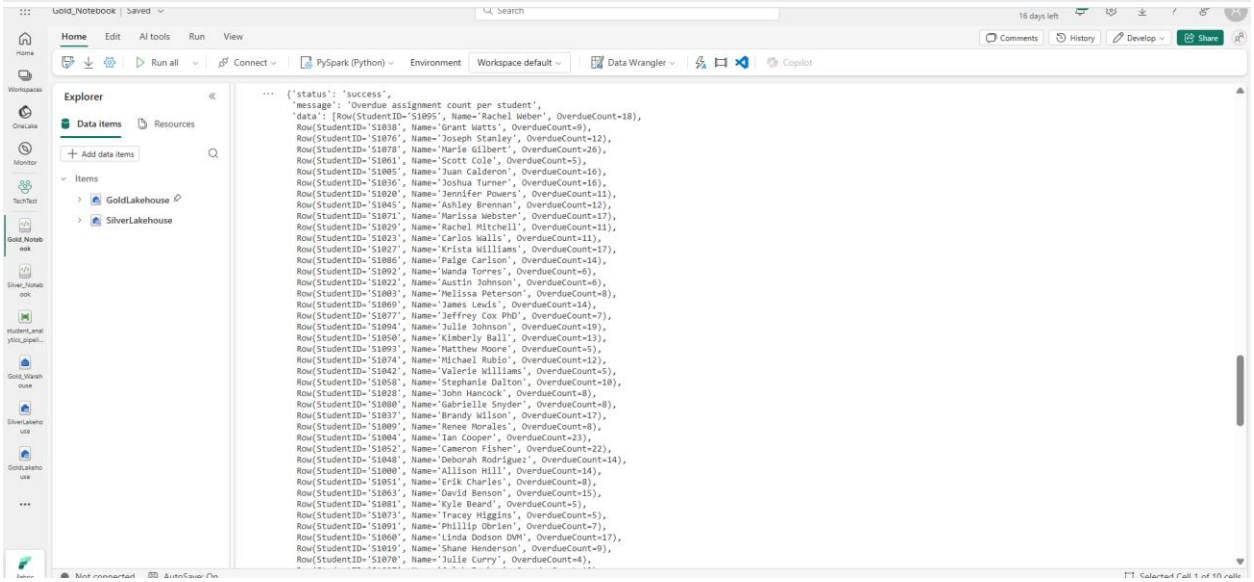


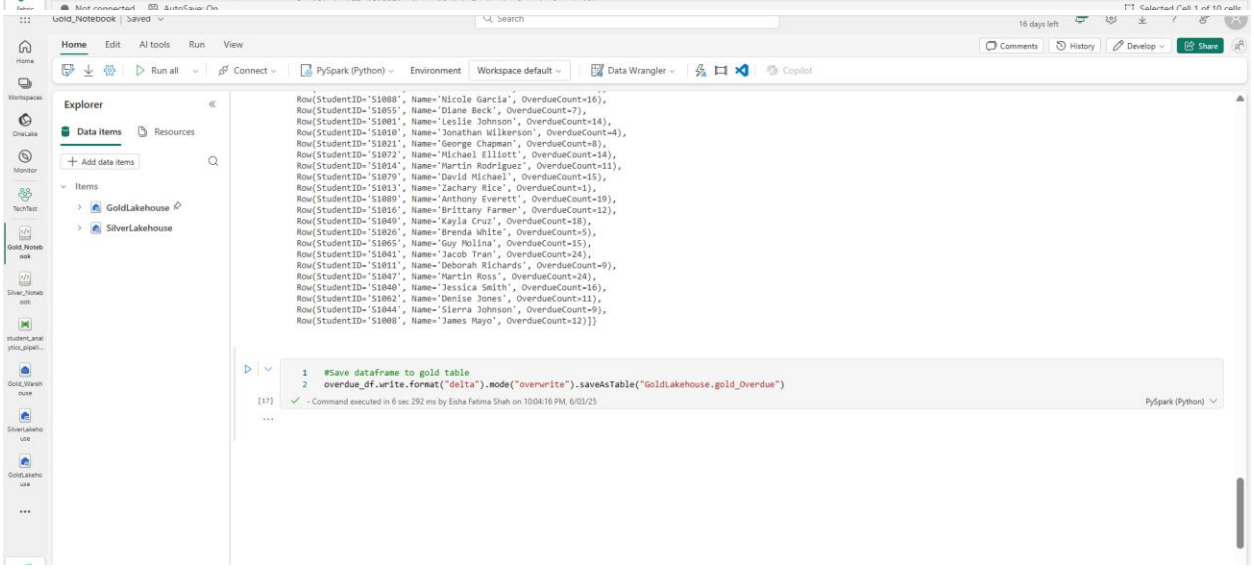



Gold_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
GoldLakehouse
SilverLakehouse
Analytical queries
# Query 1: Students who have not submitted Assignment_101
not_submitted_df = silver_df \
.filter((col("AssignmentID") == "A101") & col("SubmissionID").isNull()) \
.select("StudentID", "Name", "Email")
# Formatted JSON Output
not_submitted_result = {
"status": "success",
"message": "Students who have not submitted Assignment_101",
"data": not_submitted_df.collect()
}
display(not_submitted_result)
[12] - Command executed in 1 sec 52 ms by Eisha Fatima Shah on 10:01:09 PM, 6/03/25
PySpark (Python)
{'status': 'success',
'message': 'Students who have not submitted Assignment_101',
'data': [Row(StudentID='S1089', Name='Anthony Everett', Email='ewilson@example.com'),
Row(StudentID='S1078', Name='Marie Gilbert', Email='umarshall@example.net'),
Row(StudentID='S1048', Name='Deborah Rodriguez', Email='yreed@example.com'),
Row(StudentID='S1071', Name='Marissa Webster', Email='jamessmith@example.org'),
Row(StudentID='S1059', Name='Justin Lowery', Email='bryantlaurie@example.net'),
Row(StudentID='S1015', Name='Valerie Gill', Email='harrellkenneth@example.net'),
Row(StudentID='S1072', Name='Michael Elliott', Email='stevenscott@example.com'),
Row(StudentID='S1007', Name='Daniel Adams', Email='lynchgeorge@example.net'),
Row(StudentID='S1000', Name='Allison Hill', Email='donaldgarcia@example.net'),
Row(StudentID='S1018', Name='Rose Spence', Email='davidalvarez@example.net'),
Row(StudentID='S1082', Name='Justin Allen', Email='fbrewer@example.com'),
Row(StudentID='S1087', Name='Michael Martinez', Email='mccannangela@example.net'),
Row(StudentID='S1014', Name='Martin Rodriguez', Email='amoore@example.net'),
Row(StudentID='S1053', Name='Joshua Smith', Email='jason99@example.net'),
Row(StudentID='S1086', Name='Paige Carlson', Email='wallkenneth@example.com'),
Row(StudentID='S1029', Name='Rachel Mitchell', Email='smoore@example.org'),
Row(StudentID='S1027', Name='Krista Williams', Email='novaksara@example.org'),
Row(StudentID='S1094', Name='Julie Johnson', Email='jasmine71@example.net'),
Not connected
AutoSave: On
Selected Cell 1 of 10 cells

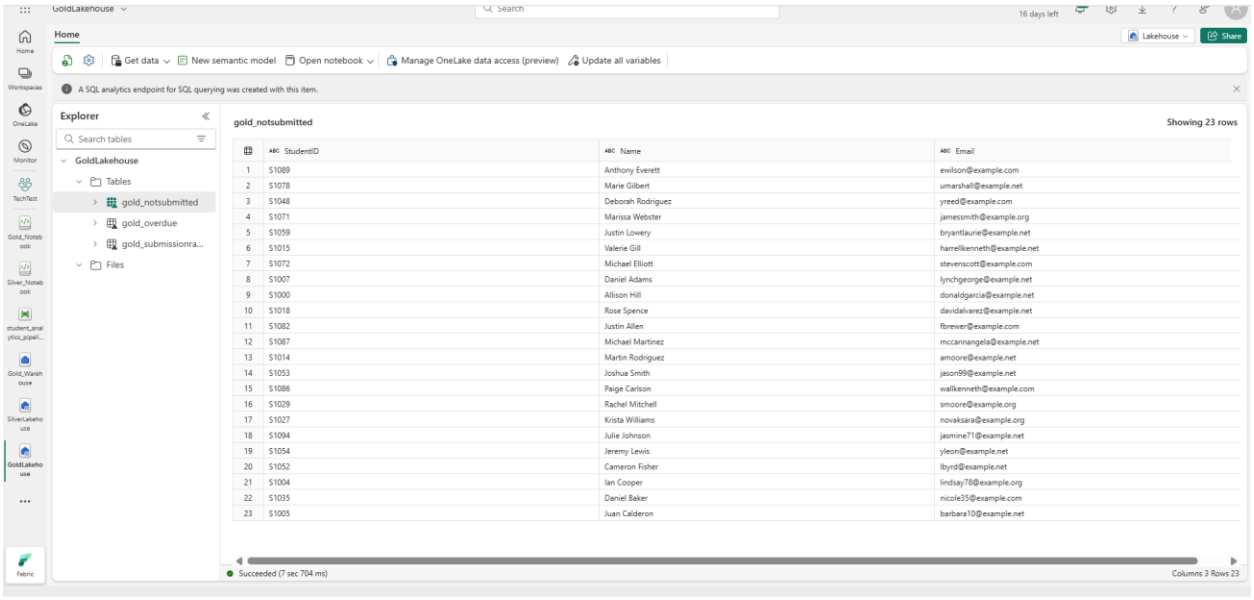

Gold_Notebook | Saved
Search
16 days left
Home
Edit
AI tools
Run
View
Comments
History
Develop
Share
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Explorer
Data items
Resources
Add data items
Items
GoldLakehouse
SilverLakehouse
#Save dataframe to gold table
not_submitted_df.write.format("delta").mode("overwrite").saveAsTable("GoldLakehouse.gold_NotSubmitted")
[15] - Command executed in 9 sec 878 ms by Eisha Fatima Shah on 10:03:13 PM, 6/03/25
PySpark (Python)
# Query 2: Submission rate per course (as a %)
# Total expected submissions per course
total_expected = silver_df.groupBy("CourseID", "CourseName").agg(count("*").alias("total_expected"))
# Actual submissions per course
actual_submissions = silver_df.filter(col("SubmissionID").isNotNull()) \
.groupBy("CourseID").agg(count("*").alias("actual_submitted"))
# Join and calculate submission rate
submission_rate_df = total_expected.join(actual_submissions, "CourseID", "left") \
.withColumn("actual_submitted", F.coalesce(col("actual_submitted"), lit(0))) \
.withColumn("SubmissionRatePercent", round((col("actual_submitted") / col("total_expected")) * 100, 2)) \
.select("CourseName", "SubmissionRatePercent")
# Format JSON Output
submission_rate_result = {
"status": "success",
"message": "Submission rate per course",
"data": submission_rate_df.collect()
}
display(submission_rate_result)
[13] - Command executed in 940 ms by Eisha Fatima Shah on 10:01:23 PM, 6/03/25
PySpark (Python)
Diagnostics
1
{'status': 'success',
'message': 'Submission rate per course',
'data': [Row(CourseName='Business-focused demand-driven flexibility', SubmissionRatePercent=8.62),
Not connected
AutoSave: On
Selected Cell 1 of 10 cells

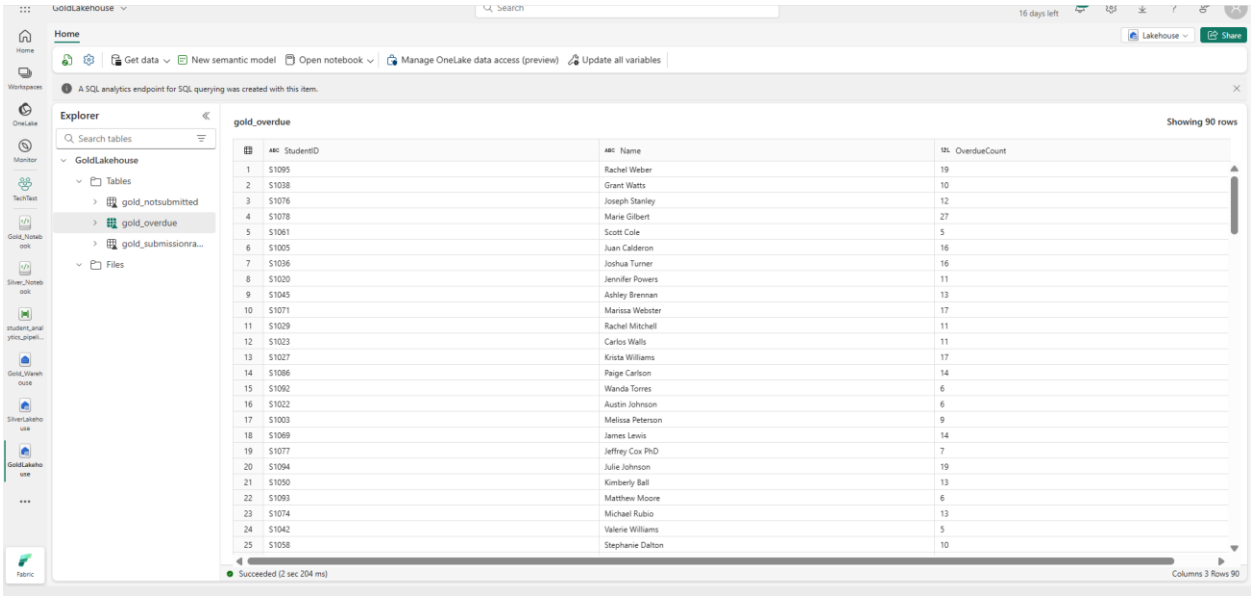

Gold_Notebook | Saved
Home
Edit
AI tools
Run
View
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Comments
History
Develop
Share
Explorer
Data items
Resources
Add data items
Items
GoldLakehouse
SilverLakehouse
Diagnostics
{'status': 'success',
'message': 'Submission rate per course',
'data': [Row(CourseName='Business-focused demand-driven flexibility', SubmissionRatePercent=8.62),
Row(CourseName='Triple-buffered cohesive function', SubmissionRatePercent=11.36),
Row(CourseName='Managed incremental solution', SubmissionRatePercent=8.54),
Row(CourseName='Organic even-keeled contingency', SubmissionRatePercent=10.7),
Row(CourseName='Customizable maximized capability', SubmissionRatePercent=12.57),
Row(CourseName='Ergonomic multi-state hierarchy', SubmissionRatePercent=7.02),
Row(CourseName='Automated logistical array', SubmissionRatePercent=12.9),
Row(CourseName='Phased grid-enabled installation', SubmissionRatePercent=8.87)]}
#Save dataframe to gold table
submission_rate_df.write.format("delta").mode("overwrite").saveAsTable("GoldLakehouse.gold_SubmissionRate")
[16] - Command executed in 5 sec 314 ms by Esha Fatima Shah on 10:03:39 PM, 6/03/25
PySpark (Python)
Diagnostics
# Query 3: Overdue assignment count per student
overdue_df = silver_df \
.filter((col("SubmissionID").isNull()) & (col("DueDate") < current_date())) \
.groupBy("StudentID", "Name") \
.agg(count("*").alias("OverdueCount"))
# Format JSON Output
overdue_result = {
"status": "success",
"message": "Overdue assignment count per student",
"data": overdue_df.collect()
}
display(overdue_result)
[14] - Command executed in 1 sec 11 ms by Esha Fatima Shah on 10:01:33 PM, 6/03/25
PySpark (Python)
Not connected
AutoSave: On
Selected Cell 1 of 10 cells

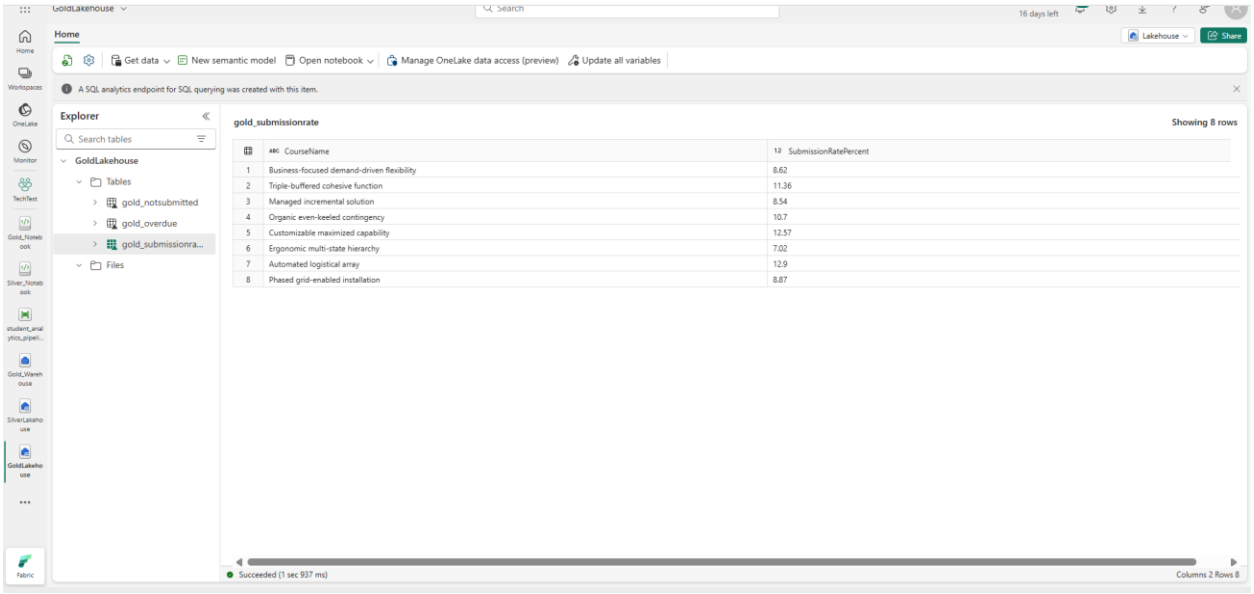

Gold_Notebook | Saved
Home
Edit
AI tools
Run
View
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Comments
History
Develop
Share
Explorer
Data items
Resources
Add data items
Items
GoldLakehouse
SilverLakehouse
{'status': 'success',
'message': 'Overdue assignment count per student',
'data': [Row(StudentID='S1095', Name='Rachel Weber', OverdueCount=18),
Row(StudentID='S1038', Name='Grant Watts', OverdueCount=9),
Row(StudentID='S1076', Name='Joseph Stanley', OverdueCount=12),
Row(StudentID='S1078', Name='Marie Gilbert', OverdueCount=26),
Row(StudentID='S1061', Name='Scott Cole', OverdueCount=5),
Row(StudentID='S1005', Name='Juan Calderon', OverdueCount=16),
Row(StudentID='S1036', Name='Joshua Turner', OverdueCount=16),
Row(StudentID='S1020', Name='Jennifer Powers', OverdueCount=11),
Row(StudentID='S1045', Name='Ashley Brennan', OverdueCount=12),
Row(StudentID='S1071', Name='Marissa Webster', OverdueCount=17),
Row(StudentID='S1029', Name='Rachel Mitchell', OverdueCount=11),
Row(StudentID='S1023', Name='Carlos Walls', OverdueCount=11),
Row(StudentID='S1027', Name='Krista Williams', OverdueCount=17),
Row(StudentID='S1086', Name='Paige Carlson', OverdueCount=14),
Row(StudentID='S1092', Name='Wanda Torres', OverdueCount=6),
Row(StudentID='S1022', Name='Austin Johnson', OverdueCount=6),
Row(StudentID='S1003', Name='Melissa Peterson', OverdueCount=8),
Row(StudentID='S1069', Name='James Lewis', OverdueCount=14),
Row(StudentID='S1077', Name='Jeffrey Cox PhD', OverdueCount=7),
Row(StudentID='S1094', Name='Julie Johnson', OverdueCount=19),
Row(StudentID='S1050', Name='Kimberly Ball', OverdueCount=13),
Row(StudentID='S1093', Name='Matthew Moore', OverdueCount=5),
Row(StudentID='S1074', Name='Michael Rubio', OverdueCount=12),
Row(StudentID='S1042', Name='Valerie Williams', OverdueCount=5),
Row(StudentID='S1058', Name='Stephanie Dalton', OverdueCount=10),
Row(StudentID='S1028', Name='John Hancock', OverdueCount=8),
Row(StudentID='S1080', Name='Gabrielle Snyder', OverdueCount=8),
Row(StudentID='S1037', Name='Brandy Wilson', OverdueCount=17),
Row(StudentID='S1009', Name='Renee Morales', OverdueCount=8),
Row(StudentID='S1004', Name='Ian Cooper', OverdueCount=23),
Row(StudentID='S1052', Name='Cameron Fisher', OverdueCount=22),
Row(StudentID='S1048', Name='Deborah Rodriguez', OverdueCount=14),
Row(StudentID='S1000', Name='Allison Hill', OverdueCount=14),
Row(StudentID='S1051', Name='Erik Charles', OverdueCount=8),
Row(StudentID='S1063', Name='David Benson', OverdueCount=15),
Row(StudentID='S1081', Name='Kyle Beard', OverdueCount=5),
Row(StudentID='S1073', Name='Tracey Higgins', OverdueCount=5),
Row(StudentID='S1091', Name='Phillip Obrien', OverdueCount=7),
Row(StudentID='S1060', Name='Linda Dodson DVM', OverdueCount=17),
Row(StudentID='S1019', Name='Shane Henderson', OverdueCount=9),
Row(StudentID='S1070', Name='Julie Curry', OverdueCount=4),
Not connected
AutoSave: On
Selected Cell 1 of 10 cells


Gold_Notebook | Saved
Home
Edit
AI tools
Run
View
Run all
Connect
PySpark (Python)
Environment
Workspace default
Data Wrangler
Copilot
Comments
History
Develop
Share
Explorer
Data items
Resources
Add data items
Items
GoldLakehouse
SilverLakehouse
Row(StudentID='S1088', Name='Nicole Garcia', OverdueCount=16),
Row(StudentID='S1055', Name='Diane Beck', OverdueCount=7),
Row(StudentID='S1001', Name='Leslie Johnson', OverdueCount=14),
Row(StudentID='S1010', Name='Jonathan Wilkerson', OverdueCount=4),
Row(StudentID='S1021', Name='George Chapman', OverdueCount=8),
Row(StudentID='S1072', Name='Michael Elliott', OverdueCount=14),
Row(StudentID='S1014', Name='Martin Rodriguez', OverdueCount=11),
Row(StudentID='S1079', Name='David Michael', OverdueCount=15),
Row(StudentID='S1013', Name='Zachary Rice', OverdueCount=1),
Row(StudentID='S1089', Name='Anthony Everett', OverdueCount=19),
Row(StudentID='S1016', Name='Brittany Farmer', OverdueCount=12),
Row(StudentID='S1049', Name='Kayla Cruz', OverdueCount=18),
Row(StudentID='S1026', Name='Brenda White', OverdueCount=5),
Row(StudentID='S1065', Name='Guy Molina', OverdueCount=15),
Row(StudentID='S1041', Name='Jacob Tran', OverdueCount=24),
Row(StudentID='S1011', Name='Deborah Richards', OverdueCount=9),
Row(StudentID='S1047', Name='Martin Ross', OverdueCount=24),
Row(StudentID='S1040', Name='Jessica Smith', OverdueCount=16),
Row(StudentID='S1062', Name='Denise Jones', OverdueCount=11),
Row(StudentID='S1044', Name='Sierra Johnson', OverdueCount=9),
Row(StudentID='S1008', Name='James Mayo', OverdueCount=12)]}
#Save dataframe to gold table
overdue_df.write.format("delta").mode("overwrite").saveAsTable("GoldLakehouse.gold_Overdue")
[17] - Command executed in 6 sec 292 ms by Esha Fatima Shah on 10:04:16 PM, 6/03/25
PySpark (Python)

Gold tables in gold lakehouse:





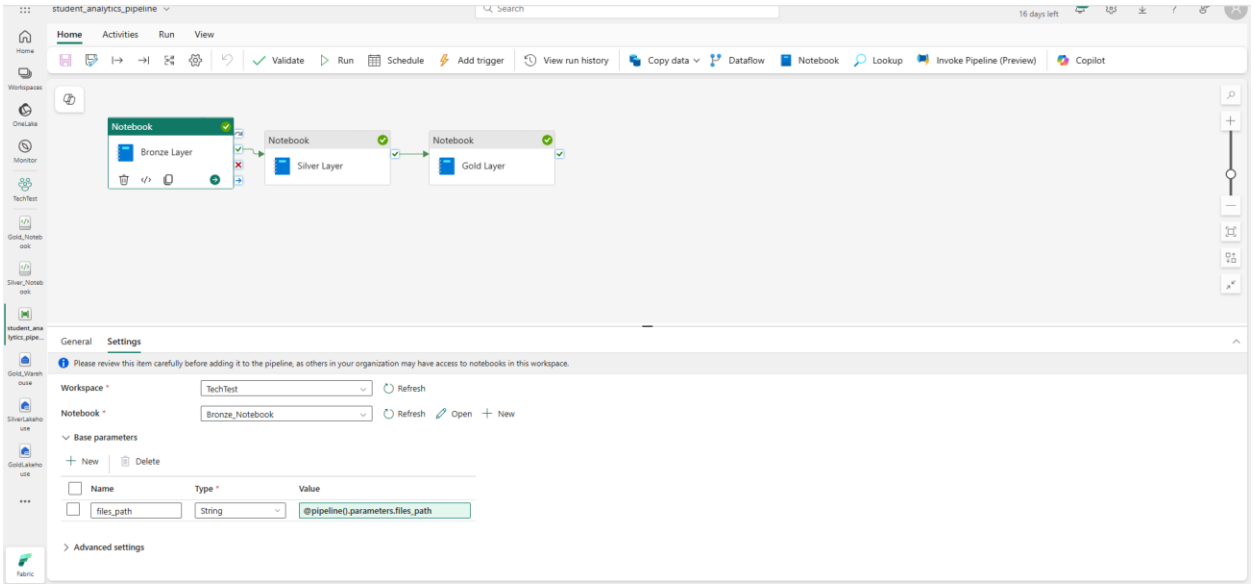


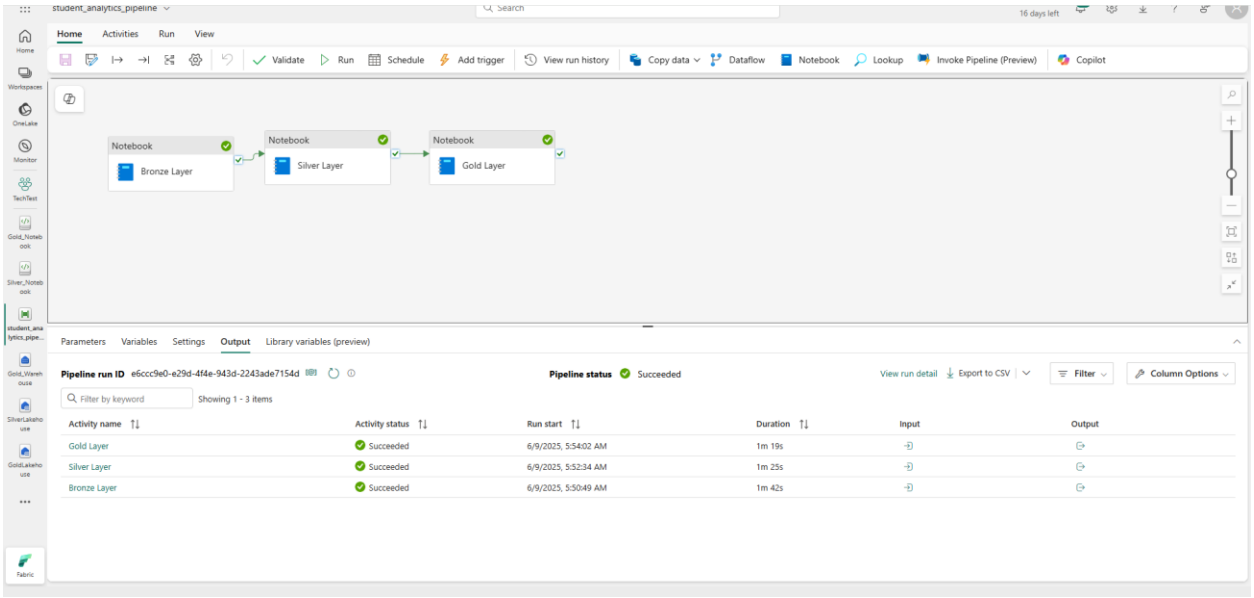
Coding the gold layer in gold notebook:

We wanted to deduce the following insights along with output:

- Students who have not submitted “Assignment_101”
- Submission rate per course (as a %)
- Overdue assignment count per student
- A consistent API-style JSON output with {status, message, data}

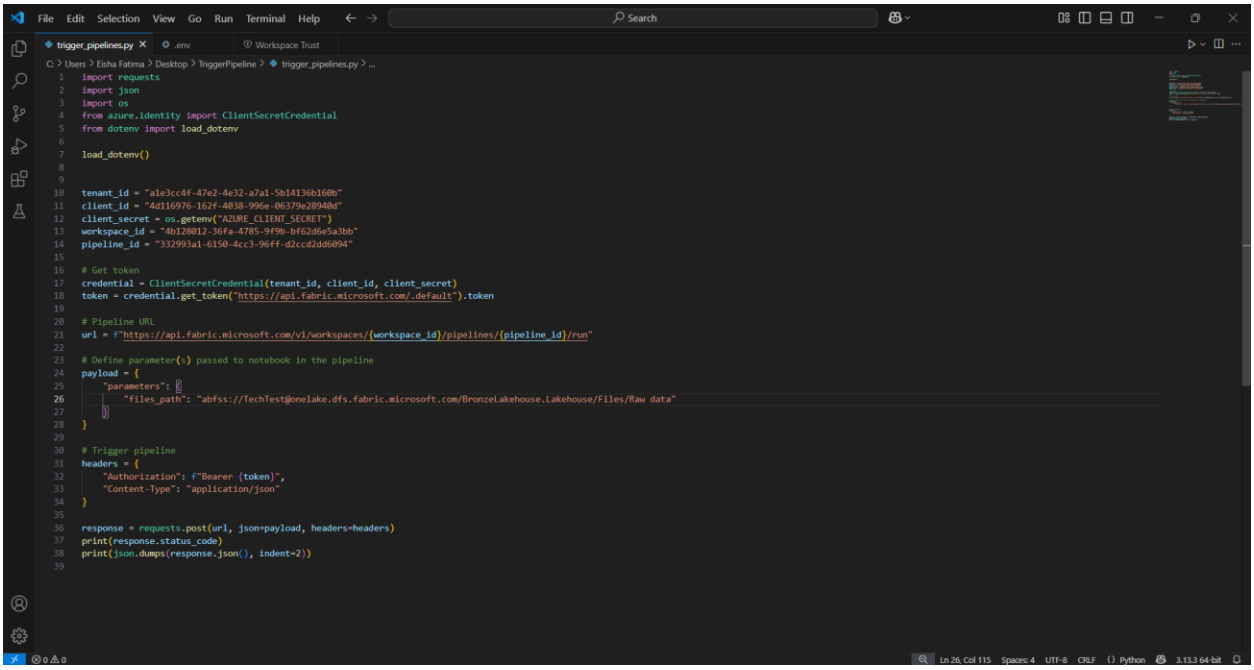
Orchestrating the flow using data pipeline in Fabric Data Factory:





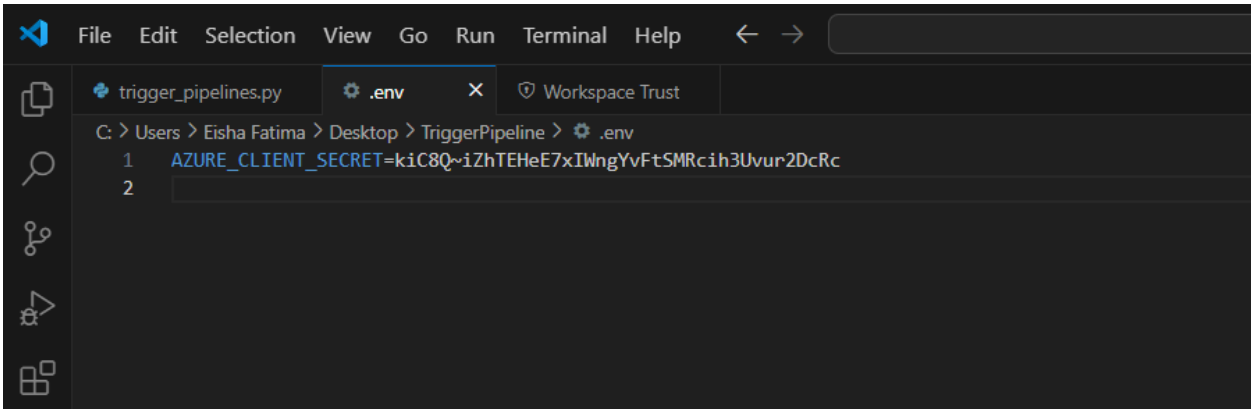
Automation using Github Actions:

Python script:

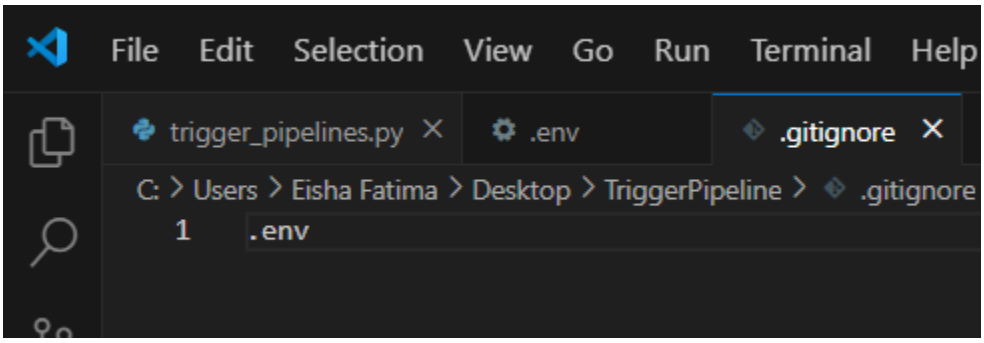


```python
import requests
import json
import os
from azure.identity import ClientSecretCredential
from dotenv import load_dotenv

load_dotenv()


tenant_id = "a1e3cc4f-47e2-4e32-a7a1-5b14136b160b"
client_id = "4d116976-162f-4038-996e-06379e28940d"
client_secret = os.getenv("AZURE_CLIENT_SECRET")
workspace_id = "4b128012-36fa-4785-9f9b-bf62d6e5a3bb"
pipeline_id = "332993a1-6150-4cc3-96ff-d2ccd2dd6094"

# Get token
credential = ClientSecretCredential(tenant_id, client_id, client_secret)
token = credential.get_token("https://api.fabric.microsoft.com/.default").token

# Pipeline URL
url = f"https://api.fabric.microsoft.com/v1/workspaces/{workspace_id}/pipelines/{pipeline_id}/run"

# Define parameter(s) passed to notebook in the pipeline
payload = {
    "parameters": {
        "files_path": "abfss://TechTest@onelake.dfs.fabric.microsoft.com/BronzeLakehouse.Lakehouse/Files/Raw data"
    }
}

# Trigger pipeline
headers = {
    "Authorization": f"Bearer {token}",
    "Content-Type": "application/json"
}

response = requests.post(url, json=payload, headers=headers)
print(response.status_code)
print(json.dumps(response.json(), indent=2))
```

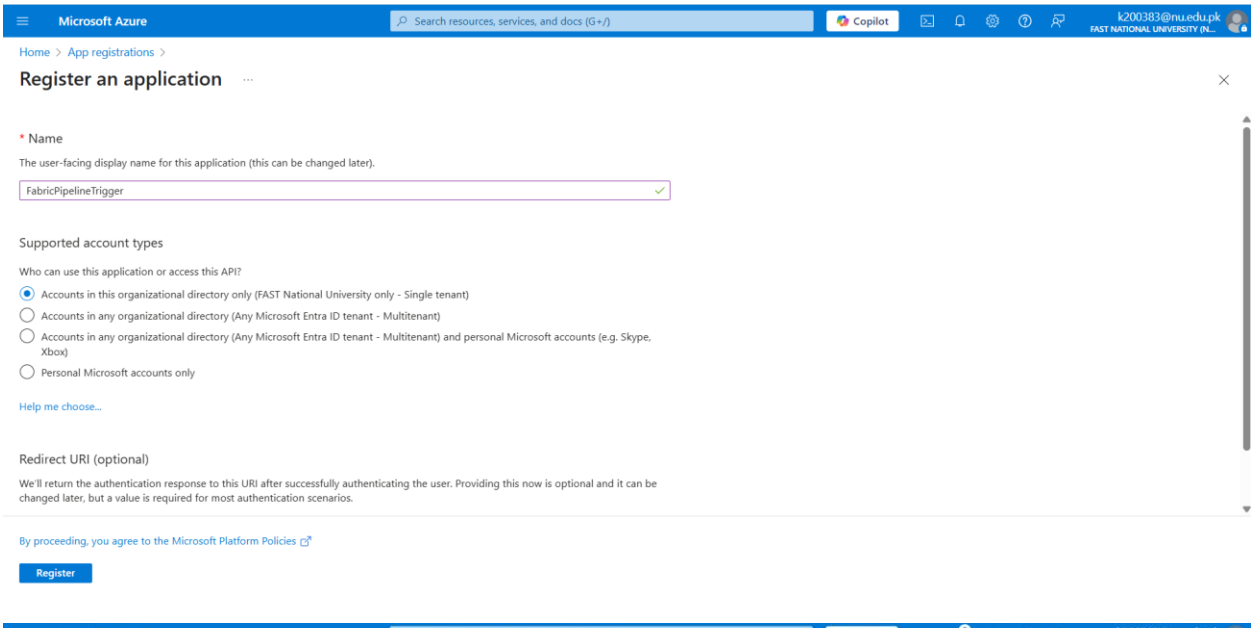
.env file:

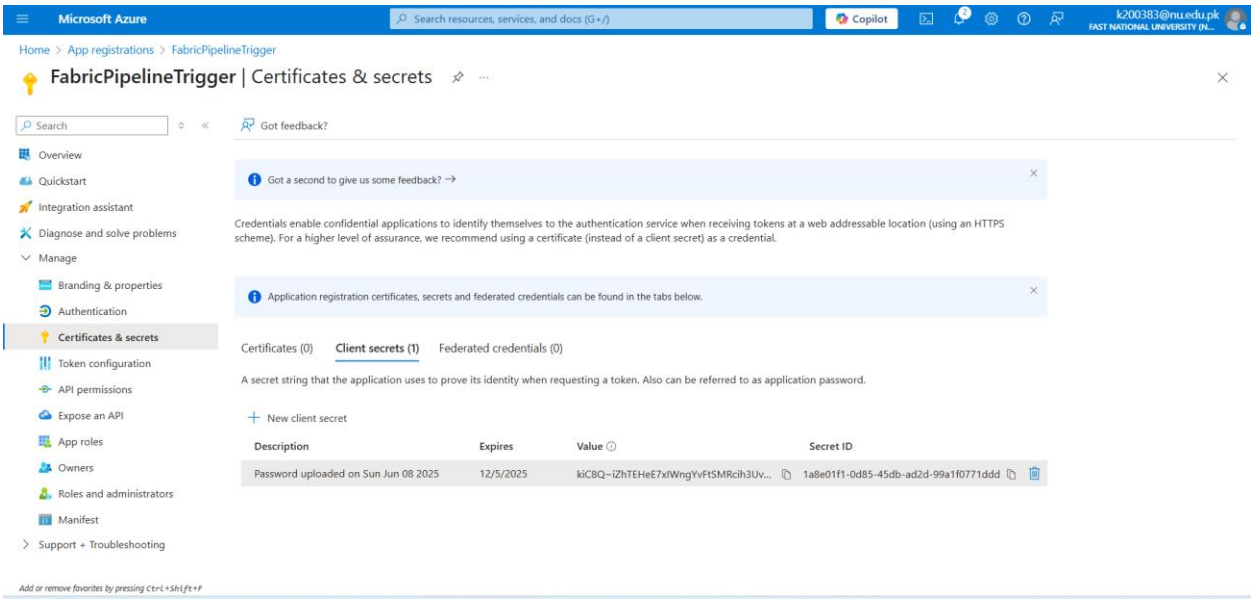


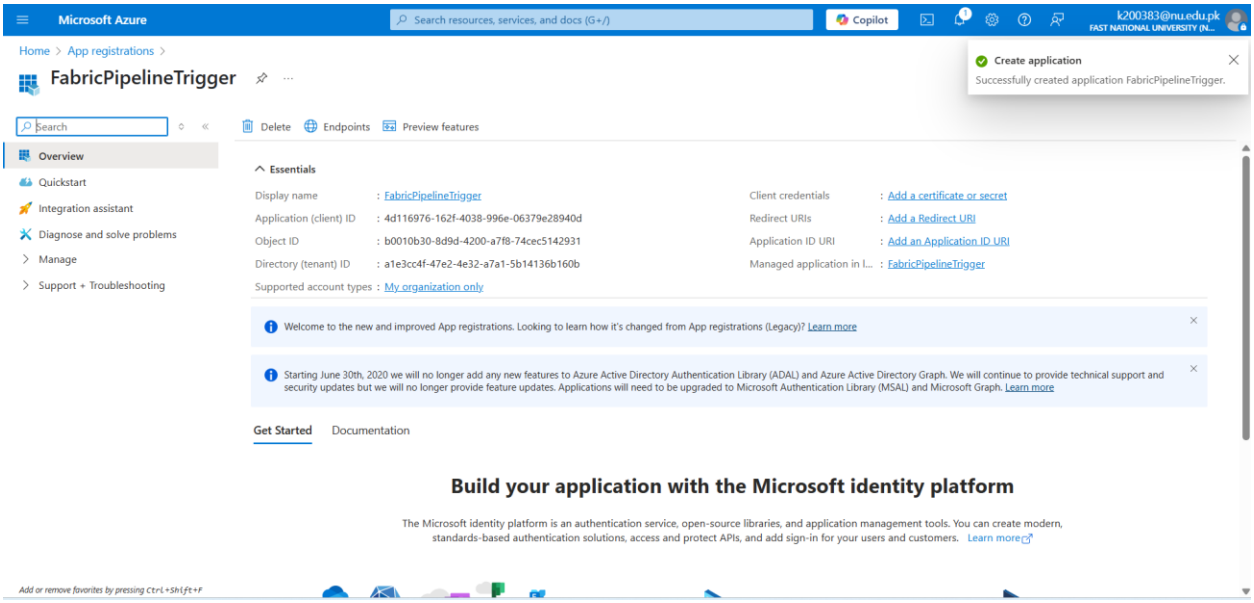
.gitignore file:



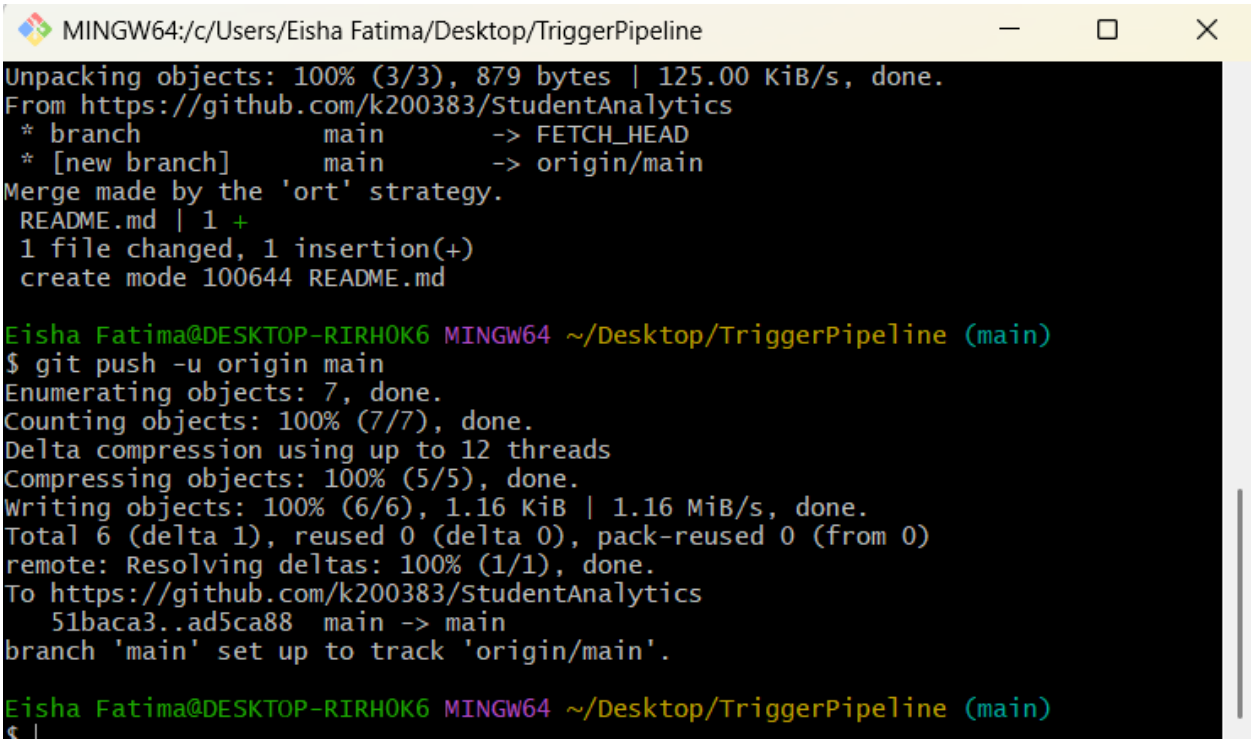
Registering Entra ID app in azure:

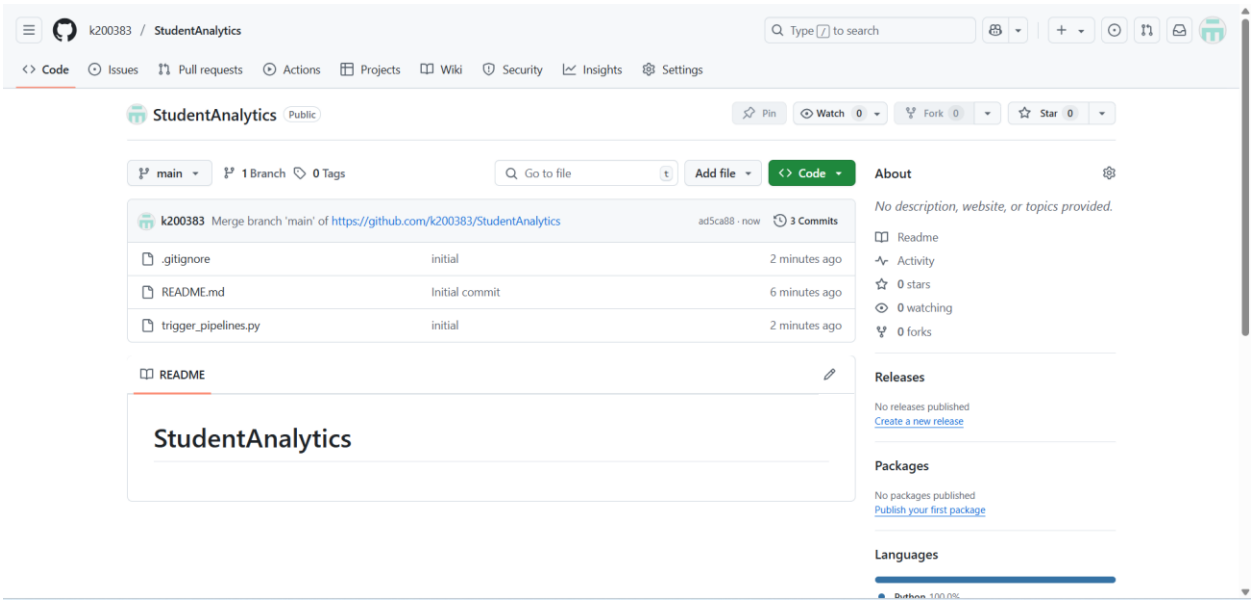




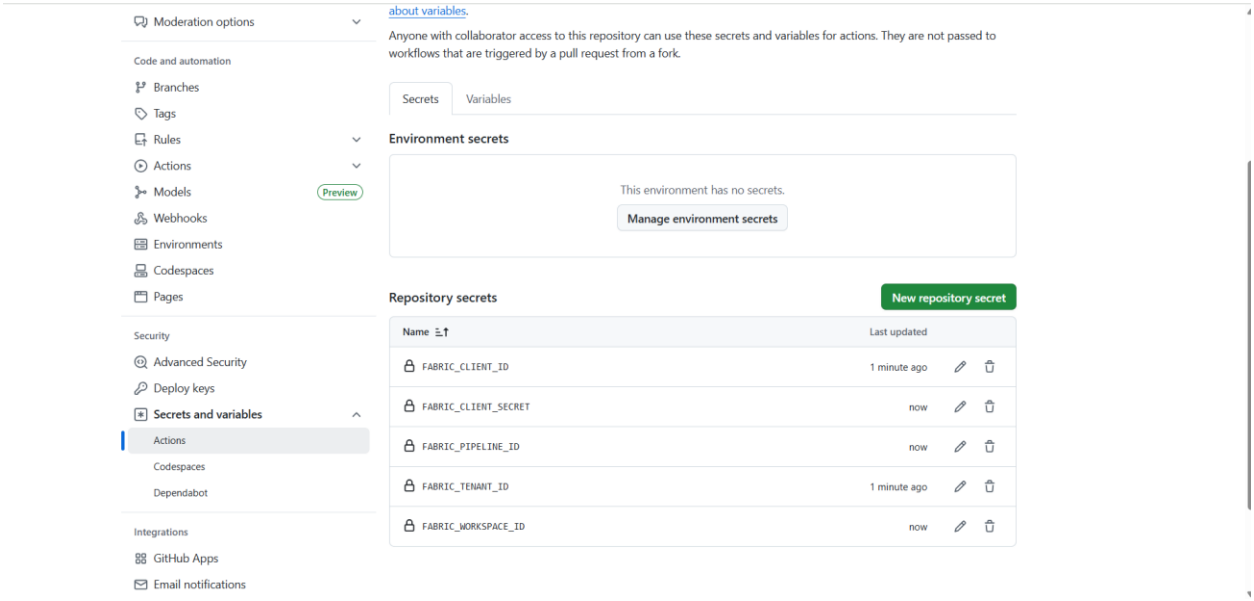


Pushing files to GitHub repository via git:

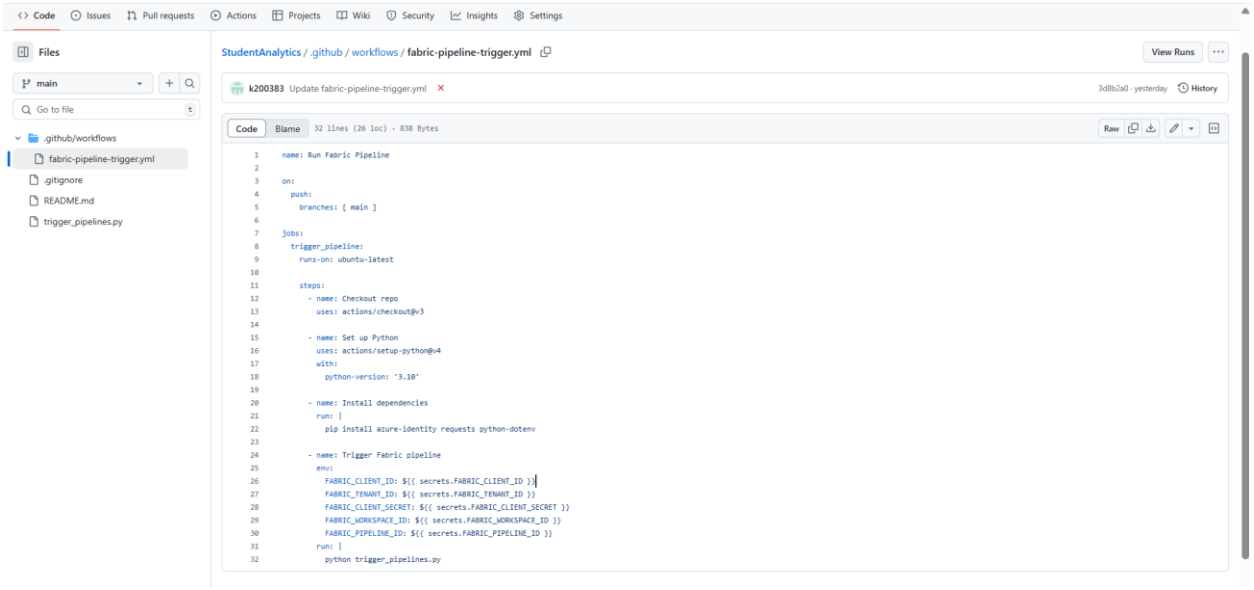


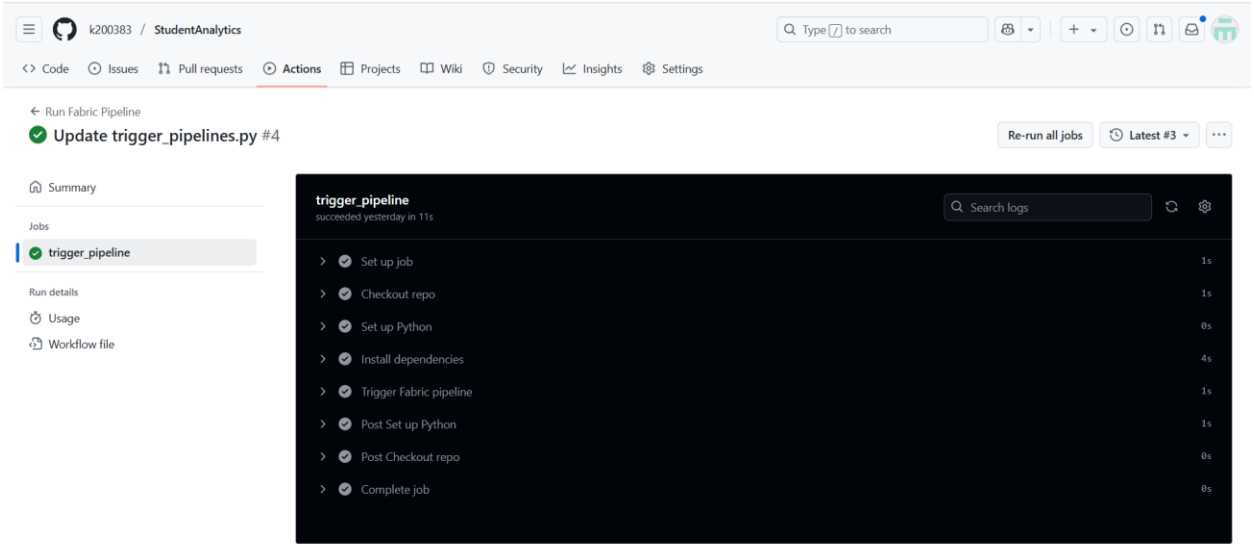


Adding Github secrets:

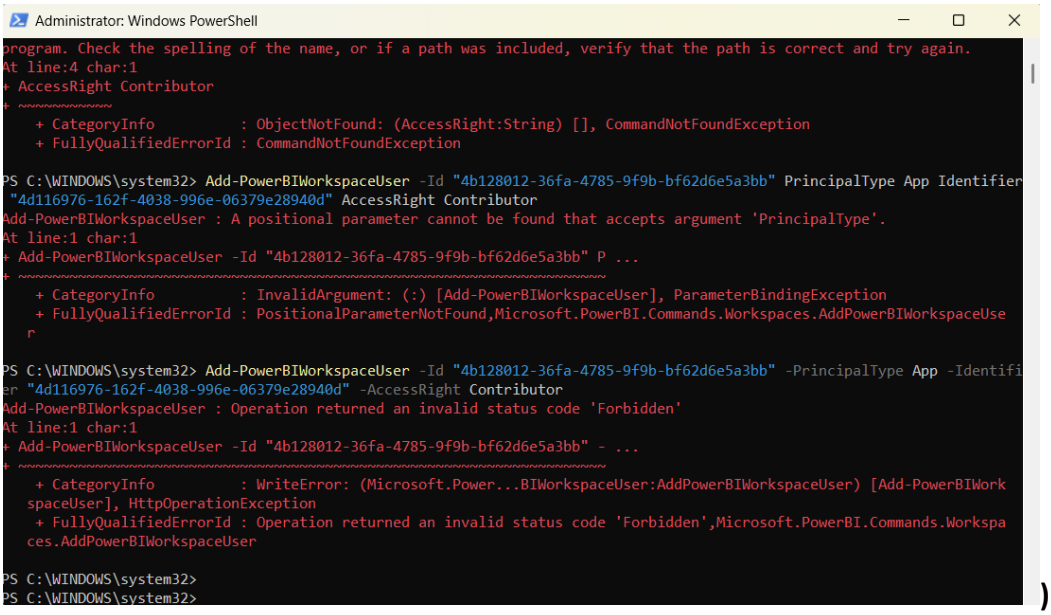


YML code for Github actions workflow:

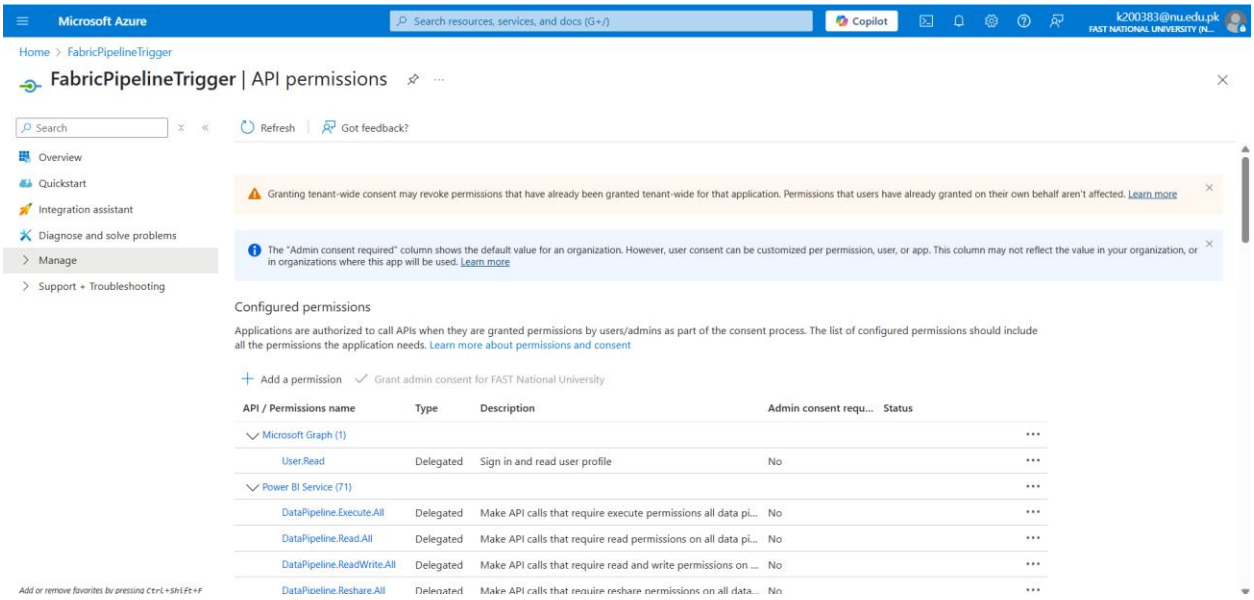




I could not access Fabric REST APIs as I am not the tenant admin (hence why I was unable to automate the fabric pipeline through the entra ID app I created):



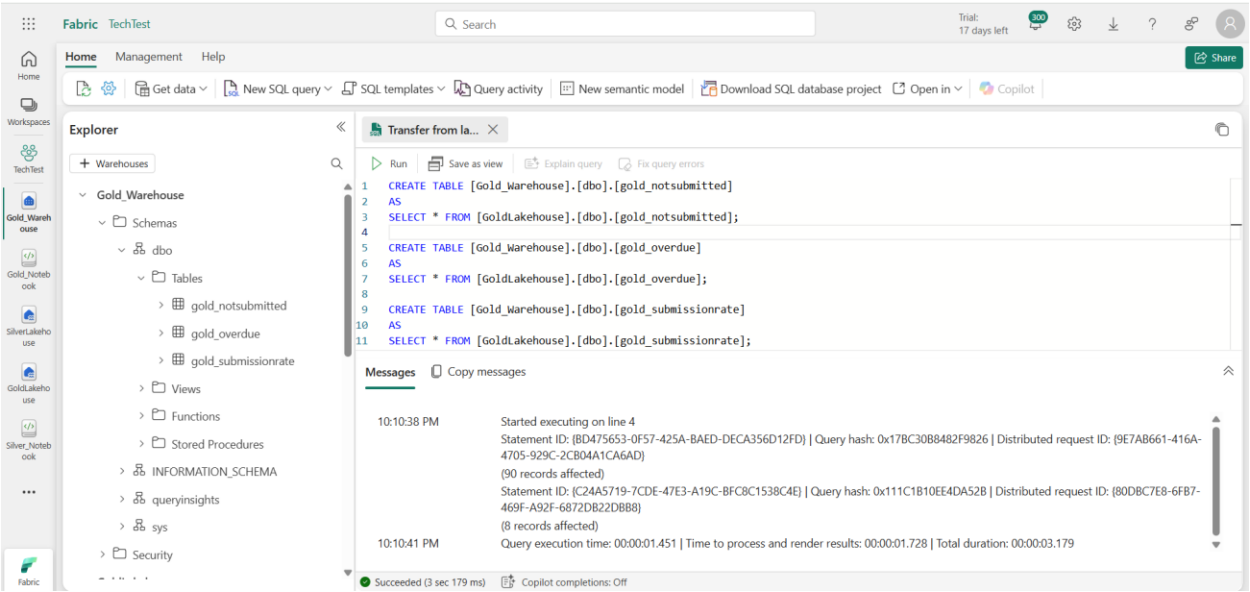
):



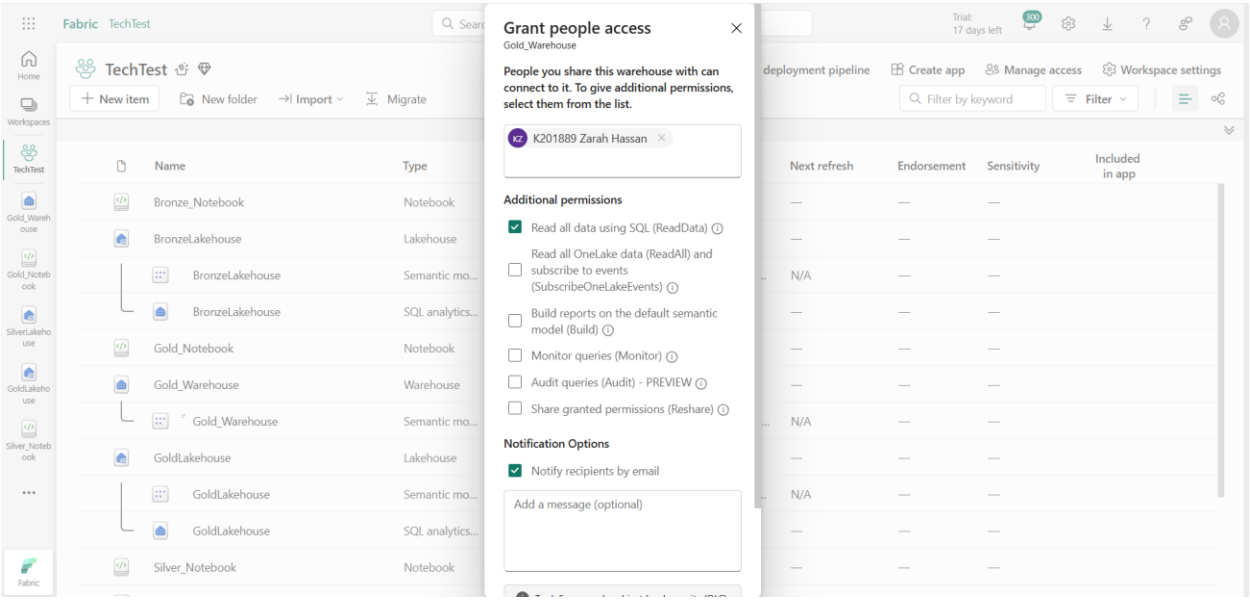
Data governance

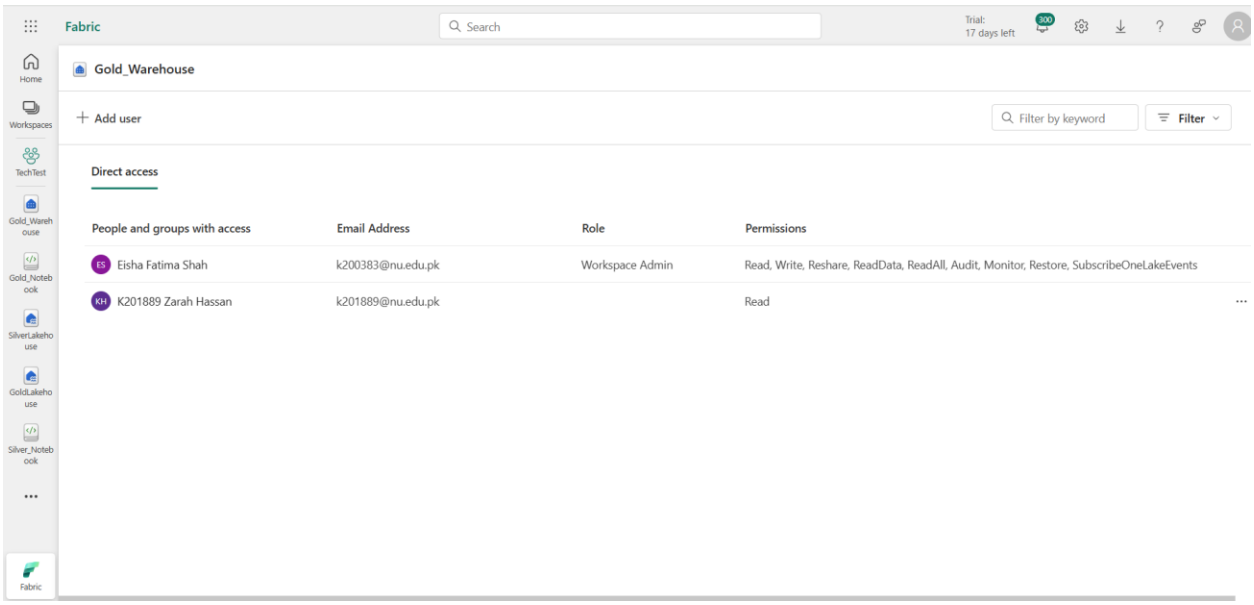
Row-level security:

Transfer data from lakehouse to warehouse:

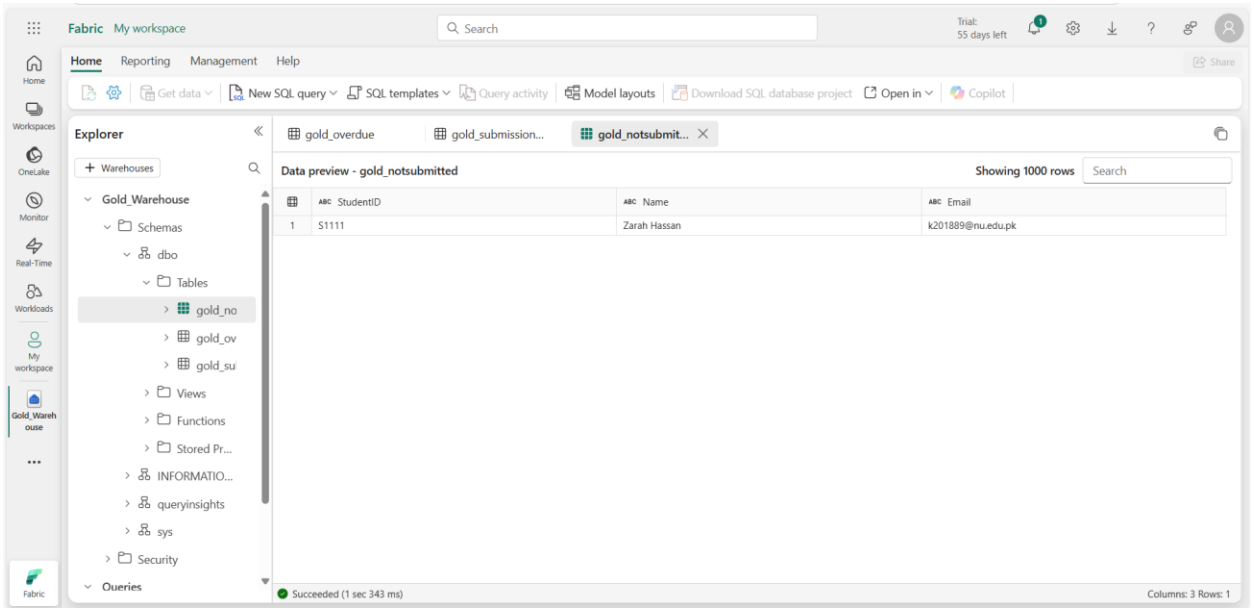


Giving only read access of warehouse to user:

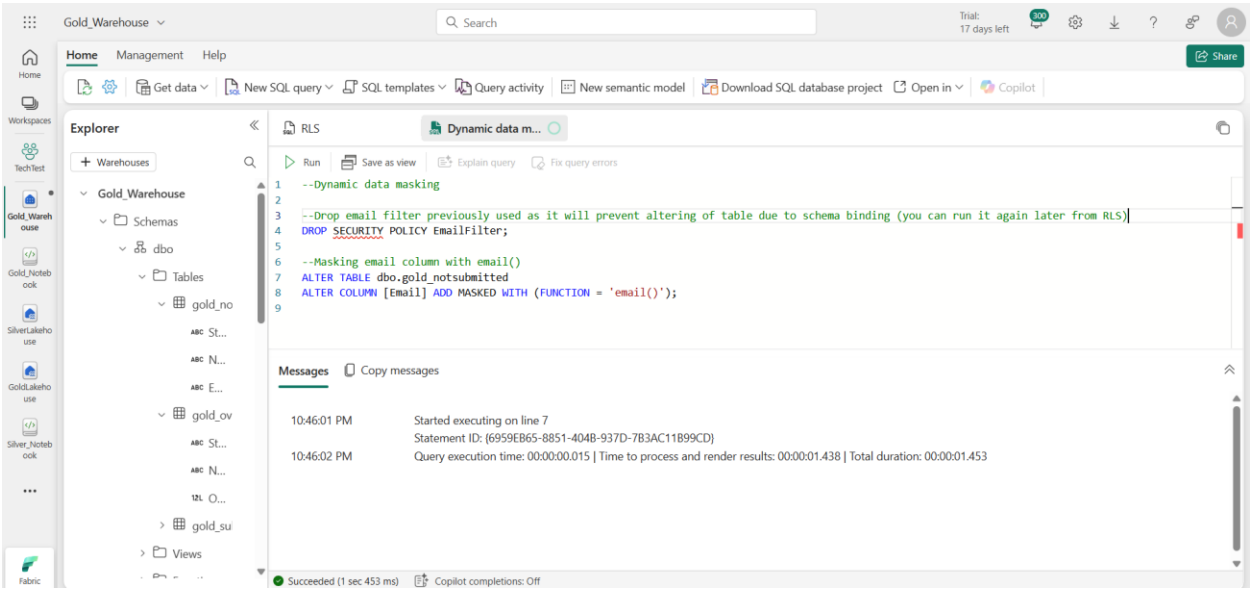


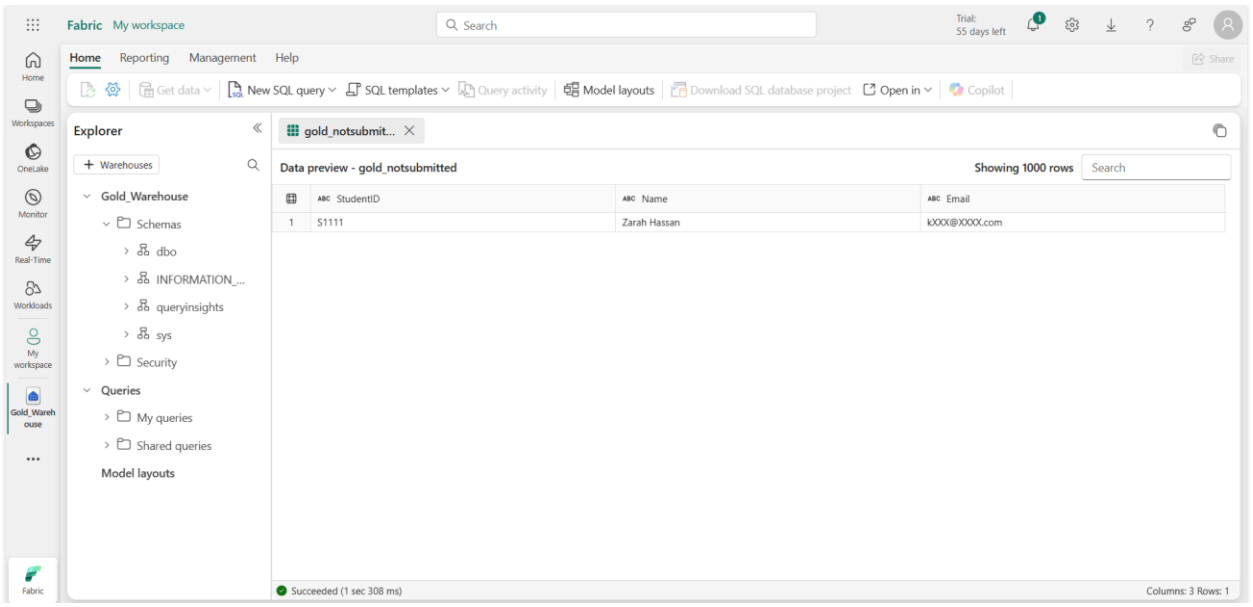


Make sure user is having only read permission to warehouse so they cannot see the data of all the warehouse tables unless access is granted to them. User can see only their email's associated information (row-level security):

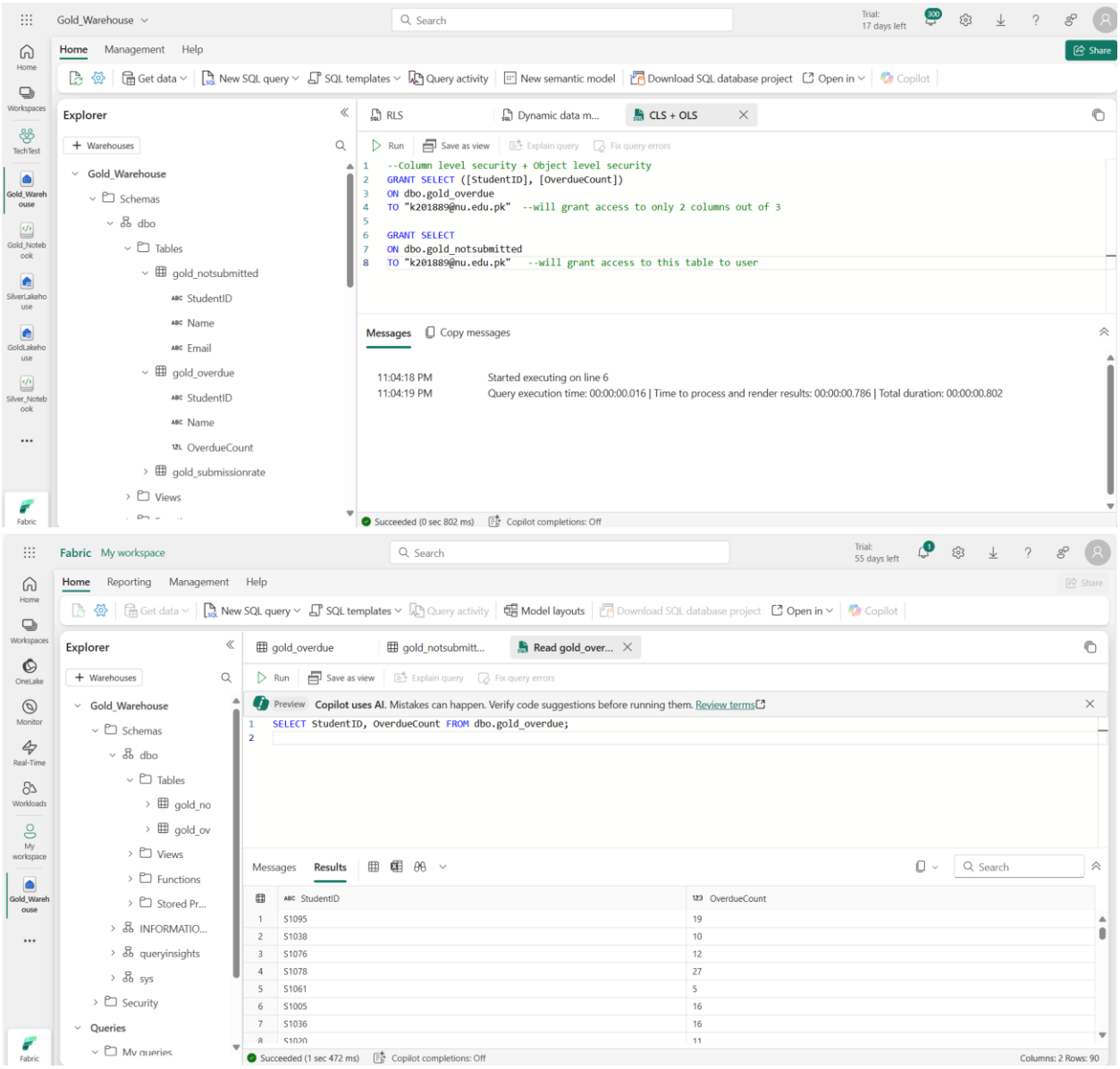


Dynamic data masking:

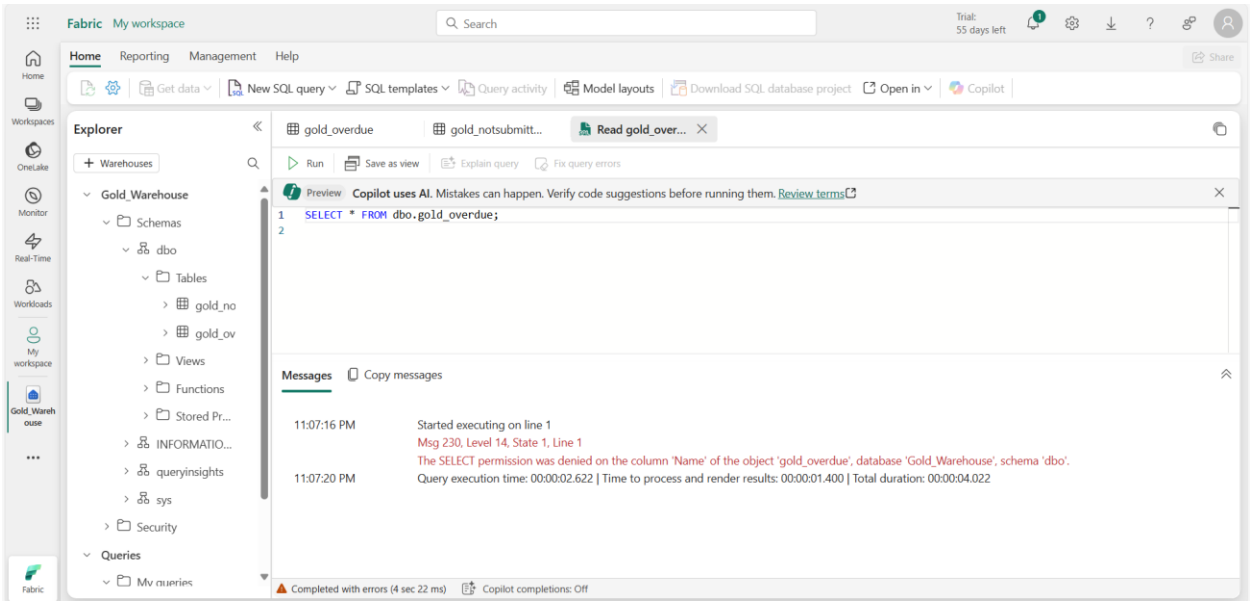




Column-level security and object-level security:



Users cannot see the names of the students but can only see the IDs.


Fabric
My workspace
Search
Trial:
55 days left
Home
Reporting
Management
Help
Share
Get data
New SQL query
SQL templates
Query activity
Model layouts
Download SQL database project
Open in
Copilot
Explorer
gold_overdue
gold_notsubmitt...
Read gold_over...
Warehouses
Run
Save as view
Explain query
Fix query errors
Gold_Warehouse
Schemas
dbo
Tables
gold_no
gold_ov
Views
Functions
Stored Pr...
INFORMATIO...
queryinsights
sys
Security
Queries
My queries
Preview
Copilot uses AI. Mistakes can happen. Verify code suggestions before running them. Review terms
1 SELECT * FROM dbo.gold_overdue;
2
Messages
Copy messages
11:07:16 PM
Started executing on line 1
Msg 230, Level 14, State 1, Line 1
The SELECT permission was denied on the column 'Name' of the object 'gold_overdue', database 'Gold_Warehouse', schema 'dbo'.
11:07:20 PM
Query execution time: 00:00:02.622 | Time to process and render results: 00:00:01.400 | Total duration: 00:00:04.022
Completed with errors (4 sec 22 ms)
Copilot completions: Off