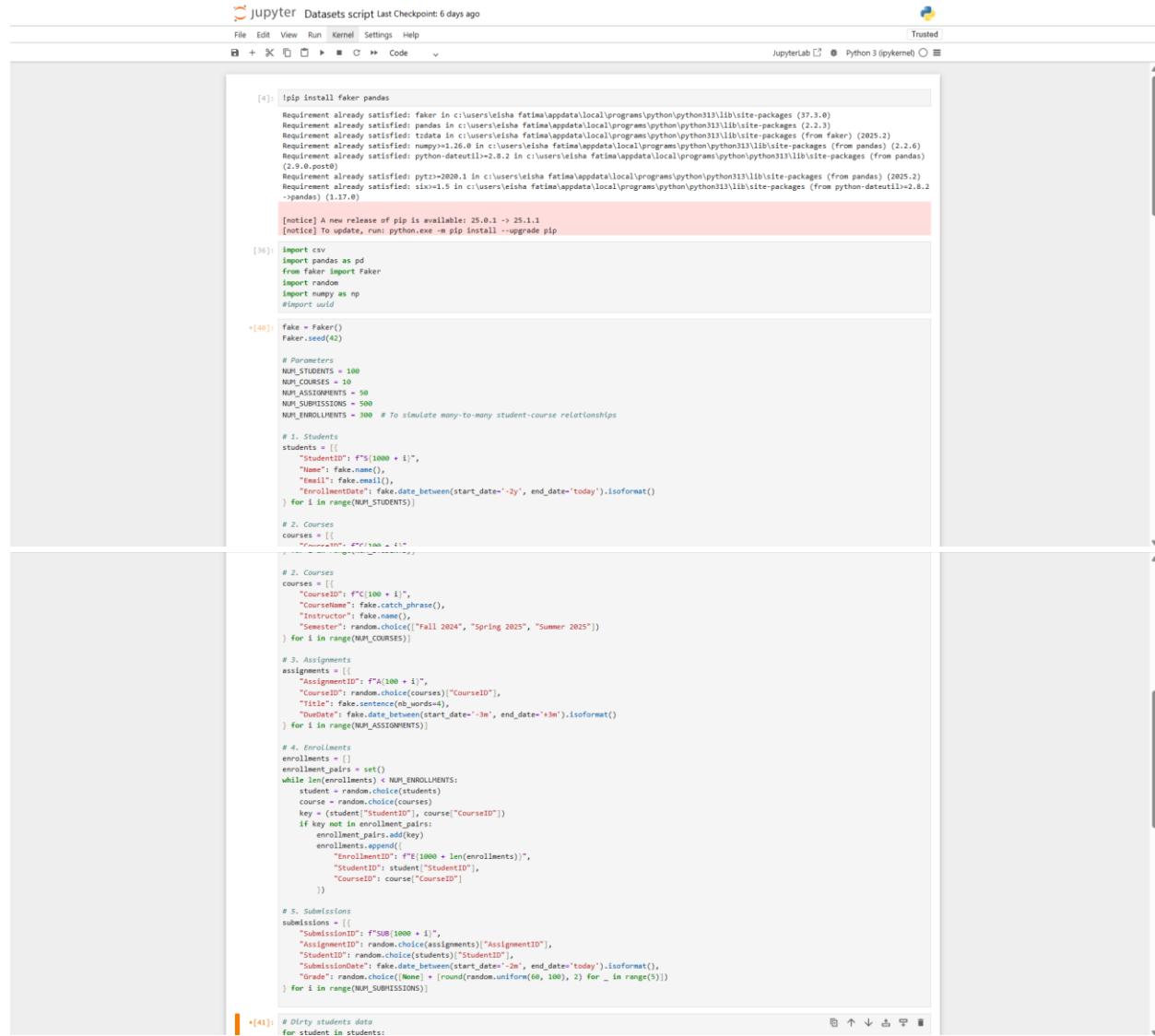


# Screenshots for Solution

## Script for simulating datasets:



The screenshot shows a Jupyter Notebook interface with a single code cell containing Python code. The code uses the Faker library to generate data for students, courses, assignments, and submissions. It defines parameters like NUM\_STUDENTS, NUM\_COURSES, etc., and then creates lists of student, course, assignment, and submission dictionaries. The code includes imports for csv, pandas, Faker, random, and numpy. It also handles pip upgrades and seed initialization.

```
[4]: !pip install faker pandas
Requirement already satisfied: faker in c:\users\elisha fatima\appdata\local\programs\python\python313\lib\site-packages (37.3.0)
Requirement already satisfied: pandas >=1.0.0 in c:\users\elisha fatima\appdata\local\programs\python\python313\lib\site-packages (from pandas) (1.5.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\elisha fatima\appdata\local\programs\python\python313\lib\site-packages (from pandas) (1.26.3)
Requirement already satisfied: python-dateutil<2.8.2 in c:\users\elisha fatima\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\elisha fatima\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: six<1.5 in c:\users\elisha 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 -m pip install --upgrade pip

[46]: import csv
import pandas as pd
from faker import Faker
import random
import numpy as np
#Import uid

[48]: 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"%{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"C100 + 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"A100 + 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"E1000 + len(enrollments)",
            "StudentID": student["StudentID"],
            "CourseID": course["CourseID"]
        ]))

# 5. Submissions
submissions = [
    {
        "SubmissionID": f"SUB1000 + 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() < 0.05:
        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_dups = int(len(data) * percent)
    duplicates = random.choices(data, k=num_dups)
    return data + duplicates

# Add duplicates (approx 5K 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)

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

# Add duplicates (approx 5K 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       | nhaldgarcia@example.org     | 2025-04-25     |
| 2  | S1001     | Leslie Johnson     | lsonwilliam@example.org     | 2024-08-11     |
| 3  | S1002     | Matthew Gardner    | mgardner@example.org        | 2025-01-12     |
| 4  | S1003     | Melissa Peterson   | mpeterson@example.org       | 2023-08-14     |
| 5  | S1004     | Ian Cooper         | iancooper@example.org       | 2025-05-19     |
| 6  | S1005     | Juan Calderon      | jcalderon@example.net       | 2023-11-27     |
| 7  | S1006     | Victoria Wyatt     | victoriawyatt@example.com   | 2023-10-07     |
| 8  | S1007     | Daniel Adams       | dcameron@example.net        | 2023-12-10     |
| 9  | S1008     | James Mayo         | jcameron@example.org        | 2023-11-27     |
| 10 | S1009     | Renee Morales      | clarkehernandez@example.net | 2024-02-02     |
| 11 | S1010     | Jonathan Wilkinson | jthomas@example.org         | 2025-01-01     |
| 12 | S1011     | Deborah Richards   | icos@example.net            | 2024-06-20     |
| 13 | S1012     | Anthony Rodriguez  | ahenderson@example.org      | 2024-03-27     |
| 14 | S1013     | Zachary Rice       | azacharyrice@example.net    | 2025-01-06     |
| 15 | S1014     | Martin Rodriguez   | amoonoo@example.net         | 2025-04-13     |
| 16 | S1015     | Valerie Gill       | valeriegill@example.net     | 2024-07-01     |
| 17 | S1016     | Brittany Farmer    | bgethacay@example.org       | 2025-05-24     |
| 18 | S1017     | William Roman      | wthompson@example.org       | 2023-08-08     |
| 19 | S1018     | Rose Spence        | rjvadavane@example.net      | 2023-10-09     |
| 20 | S1019     | Shane Henderson    | rahcampus@example.net       | 2023-11-06     |
| 21 | S1020     | Jennifer Powers    | samuelB@example.org         | 2023-12-12     |
| 22 | S1021     | George Chapman     | lannahsmith@example.net     | 2024-10-05     |
| 23 | S1022     | Austin Johnson     | chad14@example.net          | 2024-09-14     |
| 24 | S1023     | Carlos Wallis      | esanchez@example.com        | 2023-10-28     |
| 25 | S1024     | Matthew Gomez      | uvicens@example.org         | 2024-11-07     |
| 26 | S1025     | Maria Brown        | lauren11@example.org        | 2023-10-28     |
| 27 | S1026     | Brenda White       | ewright@example.com         | 2025-02-19     |
| 28 | S1027     | Krista Williams    | novakara@example.org        | 2023-10-10     |
| 29 | S1028     | John Hancock       | artnaron@example.com        | 2023-12-25     |
| 30 | S1029     | Rachel Mitchell    | smoore@example.org          | 2023-07-18     |
| 31 | S1030     | Kimberly Adkins    | msmiththomas@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         | C108     | vironment able rise study.   | 2025-06-02 |
| 5  | A104         | C109     | 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         | C109     | 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 fit better.          | 2025-06-02 |
| 16 | A115         | C108     | Kitchen reality.             | 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         | C109     | 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     | gig-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     | infected multimedia ability | Kimberly Gutierrez  | Fall 2024   |
| 6  | C105     | omic multi-state hierarchy  | Kathleen Ramos      | Spring 2025 |
| 7  | C106     | 3 demand-driven flexibility | Tanya Kim           | Summer 2025 |
| 8  | C107     | Total mobile capability     | Mark Lloyd          | Fall 2024   |
| 9  | C108     | suffered 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        | S1069     | 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     | 61.54 |
| 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     | 87.34 |
| 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         | S1028     | 2025-06-02     | 66.02 |
| 31 | SUB1030      | A148         | S1025     | 2025-06-02     | 95.08 |

## Create storage account in Azure:

The screenshot shows the 'Create a storage account' wizard in the Microsoft Azure portal. The 'Advanced' tab is selected. The 'Security' section contains several configuration options with checkboxes:

- Require secure transfer for REST API operations (checked)
- Allow enabling anonymous access on individual containers (unchecked)
- Enable storage account key access (checked)
- Default to Microsoft Entra authorization in the Azure portal (unchecked)

Below these are dropdown menus for 'Minimum TLS version' (set to 'Version 1.2') and 'Permitted scope for copy operations (preview)' (set to 'From any storage account').

The 'Hierarchical Namespace' section includes a note about enabling file and directory semantics for big data analytics workloads, followed by a checkbox for 'Enable hierarchical namespace' which is checked.

At the bottom of the screen are navigation buttons: 'Previous', 'Next', and a prominent blue 'Review + create' button.

(ticked hierachal 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):

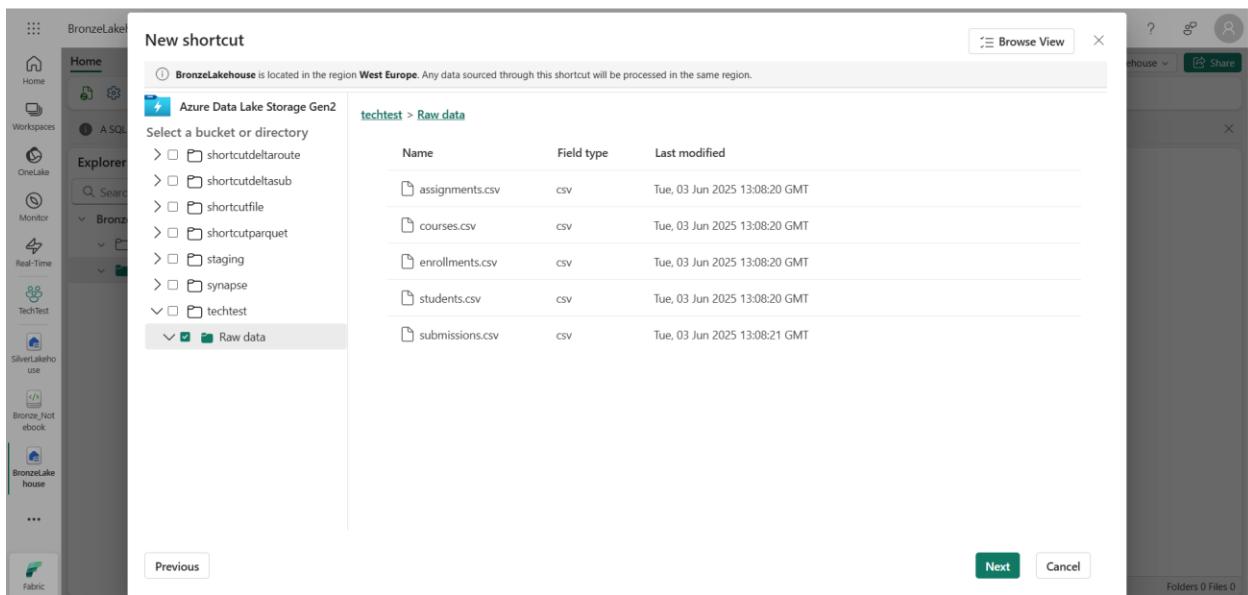
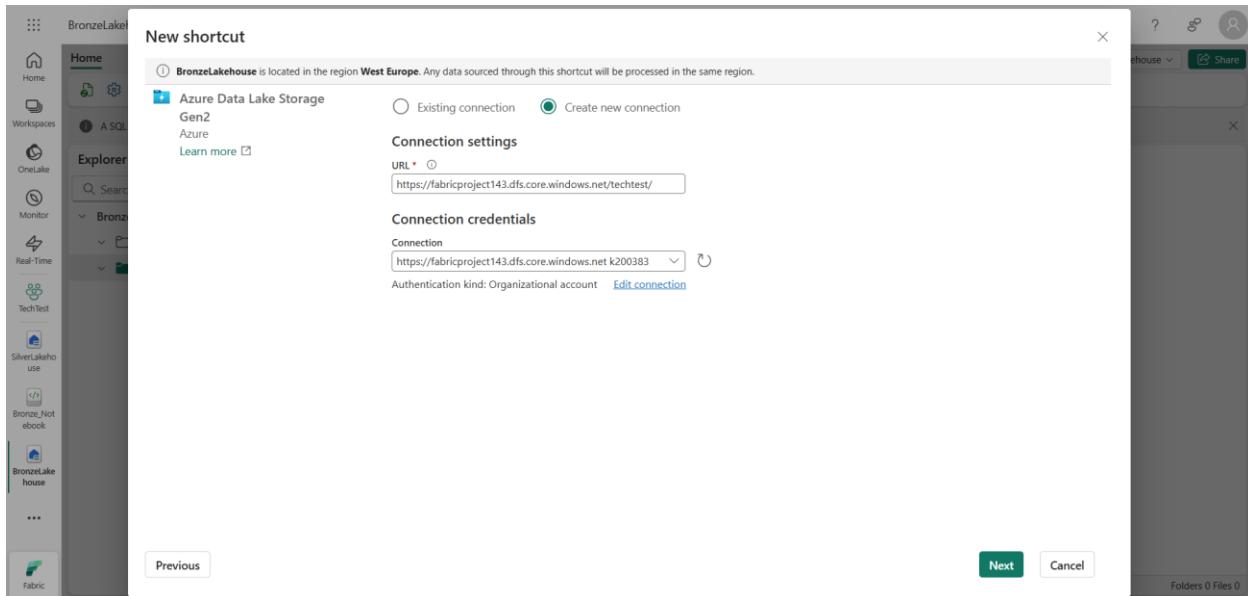
The screenshot shows the Microsoft Azure Storage Explorer interface. A container named 'techtest' is selected. The 'Overview' tab is active, displaying the following details:

- Authentication method: Access key (Switch to Microsoft Entra user account)
- Location: techtest / Raw data
- Search blobs by prefix (case-sensitive): An input field for searching blob names.
- Show deleted objects: A toggle switch that is off.

Below this is a table listing the blobs in the container:

| Name            | Modified             | Access tier    | Archive status | Blob type  | Size      | Lease state |
|-----------------|----------------------|----------------|----------------|------------|-----------|-------------|
| assignments.csv | 6/3/2025, 6:08:20 PM | Hot (Inferred) |                | Block blob | 2.33 Kib  | Available   |
| courses.csv     | 6/3/2025, 6:08:20 PM | Hot (Inferred) |                | Block blob | 685 B     | Available   |
| enrollments.csv | 6/3/2025, 6:08:20 PM | Hot (Inferred) |                | Block blob | 5.57 Kib  | Available   |
| students.csv    | 6/3/2025, 6:08:20 PM | Hot (Inferred) |                | Block blob | 5.61 Kib  | Available   |
| submissions.csv | 6/3/2025, 6:08:21 PM | Hot (Inferred) |                | Block blob | 18.44 Kib | Available   |

## Create shortcut in fabric lakehouse (bronze) from ADLS:



## Bronze lakehouse preview:

The screenshot shows the Fabric interface with the 'Fabric' tab selected. The left sidebar lists workspaces: Home, OneLake, Monitor, Real-Time, Tech Test, SilverLakehouse, Bronze\_Not notebook, and BronzeLakehouse. The main area is titled 'Home' and shows a message: 'A SQL analytics endpoint for SQL querying was created with this item.' Below this is the 'Explorer' section, which displays a tree view under 'BronzeLakehouse' with 'Tables' and 'Files'. A folder named 'Raw data' is listed under 'Files'. To the right is the 'Files' section, showing one item: 'Raw data' (Date modified: 6/3/2025, 6:13:25 PM, Type: Folder). At the bottom, a status bar indicates 'Succeeded (2 sec 796 ms)' and 'Folders 1 Files 0'.

## Ingesting data into bronze lakehouse:

The screenshot shows a Jupyter Notebook cell with the title 'Ingesting data into Bronze Lakehouse tables'. The code in the cell is as follows:

```
1 #Lakehouse files path ingested from data pipeline (parameter)
2 file_path = "samplepath"
[1] ✓ - Command executed in 433 ms by Esha Fatima Shah on 1:56:24 AM, 6/08/25
```

```
1 #Reading the CSV file (in shortcut file) into the dataframes
2 df_students = spark.read.option("header", True).csv(f"{files_path}/students.csv")
3 df_assignments = spark.read.option("header", True).csv(f"{files_path}/assignments.csv")
4 df_courses = spark.read.option("header", True).csv(f"{files_path}/courses.csv")
5 df_submissions = spark.read.option("header", True).csv(f"{files_path}/submissions.csv")
6 df_enrollments = spark.read.option("header", True).csv(f"{files_path}/enrollments.csv")
[2] ✓ - Command executed in 16 sec 946 ms by Esha Fatima Shah on 7:30:42 PM, 6/03/25
```

```
1 #Writing the CSV files to Bronze Lakehouse tables
2 df_students.write.mode("overwrite").format("delta").saveAsTable("bronze_students")
3 df_assignments.write.mode("overwrite").format("delta").saveAsTable("bronze_assignments")
4 df_courses.write.mode("overwrite").format("delta").saveAsTable("bronze_courses")
5 df_submissions.write.mode("overwrite").format("delta").saveAsTable("bronze_submissions")
6 df_enrollments.write.mode("overwrite").format("delta").saveAsTable("bronze_enrollments")
[3] ✓ - Command executed in 50 sec 566 ms by Esha Fatima Shah on 7:31:33 PM, 6/03/25
```

The status bar at the bottom indicates 'Not connected' and 'AutoSave: On'.

## Successfully created bronze tables:

The screenshot shows the Microsoft Fabric interface. The left sidebar has a 'BronzeLakehouse' section selected. In the center, under 'bronze\_assignments', there is a table view showing 52 rows of data. The columns are AssignmentID, CourseID, Title, and DueDate. The data includes various assignments like 'Meeting event strong.', 'Lead certain course.', etc., with due dates ranging from 2025-06-02 to 2025-06-03.

| AssignmentID | CourseID | Title                                   | DueDate    |
|--------------|----------|---|------------|
| A100         | C105     | Meeting event strong.                   | 2025-06-02 |
| A101         | C106     | Lead certain course.                    | 2025-06-02 |
| A102         | C109     | Second carry attorney significant take. | 2025-06-02 |
| A103         | C106     | Environment able rise study.            | 2025-06-02 |
| A104         | C108     | Tend land machine forward.              | 2025-06-02 |
| A105         | C108     | Lay focus country recently.             | 2025-06-02 |
| A106         | C104     | Do simply analysis seat.                | 2025-06-02 |
| A107         | C109     | Write society present charge.           | 2025-06-02 |
| A108         | C105     | Maybe opportunity thank.                | 2025-06-02 |
| A109         | C101     | Moment lead.                            | 2025-06-02 |
| A110         | C104     | That wide avoid sit.                    | 2025-06-02 |
| A111         | C100     | Perhaps however bag forget.             | 2025-06-02 |
| A112         | C109     | West phone thought maintain.            | 2025-06-02 |
| A113         | C106     | Pay doctor.                             | 2025-06-02 |
| A114         | C105     | Section its better.                     | 2025-06-02 |
| A115         | C108     | Kitchen really.                         | 2025-06-02 |
| A116         | C104     | Food argue.                             | 2025-06-02 |
| A117         | C102     | Issue each include radio.               | 2025-06-02 |

## Preview of silver lakehouse:

The screenshot shows the Microsoft Fabric interface with 'SilverLakehouse' selected in the sidebar. On the right, there is a 'Get data in your lakehouse' section featuring five cards: 'Upload files', 'Start with sample data', 'New shortcut', 'New Dataflow Gen2', and 'New data pipeline'.

- Upload files**: Upload data from your local machine.
- Start with sample data**: Automatically import tables filled with sample data.
- New shortcut**: Access data that resides in an external lake.
- New Dataflow Gen2**: Prep, clean, transform, and ingest data.
- New data pipeline**: Ingest data at scale and schedule data workflows.

## Coding the silver layer in notebook:

The image displays two side-by-side screenshots of the Databricks interface, showing the process of ingesting data from a Bronze Layer and performing data quality checks.

**Top Notebook: Ingesting data from Bronze Layer**

- Explorer:** Shows two Data Items: SilverLakehouse and Bronzelakehouse.
- Code:**

```
1 #Reading bronze tables
2 assignments = spark.read.format("delta").table("BronzeLakehouse.bronze_assignments")
3 courses = spark.read.format("delta").table("BronzeLakehouse.bronze_courses")
4 bronzedata = spark.read.format("delta").table("BronzeLakehouse.bronze_enrollments")
5 students = spark.read.format("delta").table("BronzeLakehouse.bronze_students")
6 submissions = spark.read.format("delta").table("BronzeLakehouse.bronze_submissions")
```

[2] ✓ - Command executed in 16 sec 249 ms by Esha Fatima Shah on 9:10:14 PM, 6/03/25 PySpark (Python) ✓
- Code:**

```
1 from pyspark.sql.functions import trim, col, lower, upper, when, to_date, initcap
2 from pyspark.sql.functions import *
3
```

[34] ✓ - Command executed in 316 ms by Esha Fatima Shah on 9:11:57 PM, 6/03/25 PySpark (Python) ✓

**Bottom Notebook: Data Quality Checks and Data Cleaning**

- Explorer:** Shows two Data Items: SilverLakehouse and Bronzelakehouse.
- Code:**

```
1 def data_quality_report(df, df_name, id_column=None):
2     print(f"\n\n Data Quality Report: {df_name}")
3     print("-" * 50)
4     print("- Schema:")
5     df.printSchema()
6
```

[6] ✓ - Not connected AutoSave: On Selected Cell 1 of 14 cells
- Code:**

```
7     print("- Record Count:", df.count())
8
9     print("- Nulls per Column:")
10    df.select({col(c).isNotNull().cast("int").alias(c) for c in df.columns}) \
11        .groupBy().sum().show()
12
13    if id_column:
14        total = df.count()
15        unique = df.select(id_column).distinct().count()
16        print(f"- Duplicates in '{id_column}': {total - unique}")
17
```

[4] ✓ - Command executed in 358 ms by Esha Fatima Shah on 9:10:15 PM, 6/03/25 PySpark (Python) ✓
- Code:**

```
1 #Cleaning students data
2 data_quality_report(students, "Students (Uncleaned)", "StudentID")
3
4 students_clean = (
5     .dropDuplicates(["StudentID"])
6     .withColumn("StudentID", trim(col("StudentID")))
7     .withColumn("Name", trim(col("Name")))
8     .withColumn("Email", trim(col("Email")))
9     .withColumn("EnrollmentDate", when(col("EnrollmentDate") == "", None).otherwise(col("EnrollmentDate"))))
10    .withColumn("EnrollmentDate", to_date("EnrollmentDate"))
11    .dropna(subset=["Email", "EnrollmentDate"])
12 )
13
14 data_quality_report(students_clean, "Students (Cleaned)", "StudentID")
15
```

[5] ✓ - Command executed in 10 sec 577 ms by Esha Fatima Shah on 9:10:25 PM, 6/03/25 PySpark (Python) ✓
- Data quality Report: Students (Uncleaned)**

```
- Schema:
root
```

Selected Cell 1 of 14 cells

Silver\_Notebook | Saved | 16 days left

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Comments History Develop Share

**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

Selected Cell 1 of 14 cells

Fabric

Not connected AutoSave: On

Silver\_Notebook | Saved | 16 days left

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Comments History Develop Share

**Explorer**

**Data items** Resources

+ Add data items

Items

- SilverLakehouse
- BronzeLakehouse

```
1 #Cleaning courses data
2 data_quality_report(courses, "Courses (Uncleaned)", "CourseID")
3
4 courses_clean ~ (
5     courses
6     .dropDuplicates("CourseID")
7     .withColumn("Instructor", when(col("Instructor").rlike("[a-zA-Z]"), initcap(trim(col("Instructor")))).otherwise(trim(col("Instructor"))))
8     .withColumn("Semester", when(col("Semester") == "Fall2024", "Fall 2024")
9         .when(col("Semester") == "SUMMER 2025", "Summer 2025")
10        .otherwise(trim(col("Semester"))))
11     .dropna(subset=["CourseID", "CourseName", "Instructor"])
12     )
13
14 data_quality_report(courses_clean, "Courses (Cleaned)", "CourseID")
15
16 ✓ - Command executed in 6 sec 760 ms by Esha Fatima Shah on 9/10/22 PM, 6/0/25
```

PySpark (Python)

Selected Cell 1 of 14 cells

Fabric

Not connected AutoSave: On

Silver\_Notebook | Saved 16 days left

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Comments History Develop Share

**Explorer**

- 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 'AssignmentID': 0

1 #cleaning assignments data
2 data\_quality\_report(assignments, "Assignments", "AssignmentID")
3
4 assignments\_clean = (
5 assignments
6 .dropDuplicates(["AssignmentID"])
7 .withColumn("AssignmentID", trim(col("AssignmentID")))
8 .withColumn("CourseID", trim(col("CourseID")))
9 .withColumn("Title", trim(col("Title")))
10 .withColumn("DueDate", to\_date(col("DueDate")))
11 .dropna(subset=["AssignmentID", "CourseID"])
12 )
13
14 data\_quality\_report(assignments\_clean, "Assignments (Cleaned)", "AssignmentID")
15

Selected Cell 1 of 14 cells

Silver\_Notebook | Saved 16 days left

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Comments History Develop Share

**Explorer**

- Duplicates in 'AssignmentID': 2

Data Quality Report: Assignments (Cleaned)

Schema:

```
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': 0

Selected Cell 1 of 14 cells

Silver\_Notebook | Saved

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Comments History Develop Share

**Explorer**

**Data Items** Resources

+ Add data items

Items

- > SilverLakehouse
- > BronzeLakehouse

```

1 #Cleaning Enrollments data
2 data_quality_report(enrollments, "Enrollments", "EnrollmentID")
3
4 enrollments_clean = (
5     enrollments
6     .dropDuplicates(["EnrollmentID"])
7     .withColumn("StudentID", trim(col("StudentID")))
8     .withColumn("CourseID", upper(trim(col("CourseID"))))
9     .dropna(subset=["StudentID", "CourseID"])
10 )
11
12 data_quality_report(enrollments_clean, "Enrollments (Cleaned)", "EnrollmentID")
13

```

[4] ✓ - Command executed in 5 sec 144 ms by Esha 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:

```

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

**PySpark (Python)**

Selected Call 1 of 14 calls

Silver\_NOTEBOOK | Saved

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Comments History Develop Share

**Explorer**

**Data Items** Resources

+ Add data items

Items

- > SilverLakehouse
- > BronzeLakehouse

```

1 #Cleaning submissions data
2 data_quality_report(submissions, "Submissions", "SubmissionID")
3
4 submissions_clean = (
5     submissions
6     .dropDuplicates(["SubmissionID"])
7     .withColumn("AssignmentID", trim(col("AssignmentID")))
8     .withColumn("StudentID", trim(col("StudentID")))
9     .withColumn("SubmissionDate", to_date(trim(col("SubmissionDate"))))
10    .withColumn("Grade", col("Grade").cast("double"))
11    .dropna(subset=["AssignmentID", "StudentID"])
12 )
13
14 data_quality_report(submissions_clean, "Submissions (Cleaned)", "SubmissionID")
15

```

[5] ✓ - Command executed in 5 sec 413 ms by Esha Fatima Shah on 9:10:48 PM, 6/03/25 PySpark (Python)

**Data Quality Report: Submissions**

Selected Call 1 of 14 calls

Silver\_Notebook | Saved | 16 days left | Comments | History | Develop | Share

Home Edit AI tools Run View

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 |  |  |  |

Selected Cell 1 of 14 cells

Not connected AutoSave: On

**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_courses_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-Assessment 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_courses_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"
35 )
36
37 # Step 5: Join with submissions
38 #All students + assignments are kept, even if there's no submission yet
39 #If student hasn't submitted, SubmissionID, SubmissionDate, Grade will be null
40 #Final silver table, from which we find missing submissions, track submission dates, calculate late submissions
41 silver_df = student_assignment_df.join(
42     submissions_clean.select("AssignmentID", "StudentID", "SubmissionID", "SubmissionDate", "Grade"),
43     on=["StudentID", "AssignmentID"],
44     how="left"
45 )
46
47 # Step 5: Join with submissions
48 #All students + assignments are kept, even if there's no submission yet
49 #If student hasn't submitted, SubmissionID, SubmissionDate, Grade will be null
50 #Final silver table, from which we find missing submissions, track submission dates, calculate late submissions
51 silver_df = student_assignment_df.join(
52     submissions_clean.select("AssignmentID", "StudentID", "SubmissionID", "SubmissionDate", "Grade"),
53     on=["StudentID", "AssignmentID"],
54     how="left"
55 )

```

Selected Cell 1 of 14 cells

**Writing dataframe into Silver Table**

```

1 silver_df.write.format("delta").mode("overwrite").saveAsTable("SilverLakehouse.silver_table")

```

[14] ✓ - Command executed in 23 sec 93 ms by Esha Fatima Shah on 9:12:51 PM, 6/03/22

PySpark (Python) ▾

Selected Cell 1 of 14 cells

## Silver table in silver lakehouse:

The screenshot shows the Azure Data Lake Storage Explorer interface. On the left, the sidebar lists various datasets: OneLake, Monitor, Run Time, Tech Test, Silver\_Notebook, student\_angry\_pupil..., Gold\_Warehouse, SilverLakehouse, Gold\_Lakehouse, and Fabric. The main area is titled 'silver\_table' under the 'SilverLakehouse' database. It displays a table with columns: StudentID, AssignmentID, Name, Email, CourseID, CourseName, Title, DueDate, SubmissionID, SubmissionDate, and Grade. The table contains 25 rows of sample data. At the bottom, a status bar indicates 'Succeeded (7 sec 547 ms)'.

The screenshot shows the Azure Data Lake Studio interface with a notebook titled 'Gold\_Notebook'. The sidebar includes icons for Home, Edit, AI tools, Run, View, Connect, PySpark (Python), Environment, Workspace default, Data Wrangler, Copilot, and Share. The notebook content includes sections for 'Ingesting data from Silver Layer' and 'Analytical queries'. In the 'Ingesting data from Silver Layer' section, a PySpark cell contains the following code:

```

1 from pyspark.sql import functions as F
2 from pyspark.sql.functions import *

```

A note below the cell says: "Command executed in 341 ms by Esha Fatima Shah on 9:55:09 PM, 6/03/25". In the 'Analytical queries' section, another PySpark cell contains:

```

1 #Reading data from silver table
2 silver_df = spark.read.format("delta").table("SilverLakehouse.silver_table")

```

A note below the cell says: "Session ready in 11 sec 620 ms. Command executed in 23 sec 254 ms by Esha Fatima Shah on 9:26:24 PM, 6/03/25". Below these cells, a third cell shows analytical query code:

```

1 # Query 1: Students who have not submitted Assignment_101
2
3 not_submitted_df = silver_df \
4     .filter((col("AssignmentID") == "A101") & col("SubmissionID").isNull()) \
5     .select("StudentID", "Name", "Email")
6
7 # Formatted JSON Output
8 not_submitted_result = [
9     {
10         "status": "Success",
11         "message": "Students who have not submitted Assignment_101",
12     }
]

```

A note below the cell says: "Selected Cell 1 of 10 cells".

The screenshot shows the Databricks interface with the following details:

- Header:** Goli\_Notebook | Saved | 16 days left | Comments | History | Develop | Share
- Left Sidebar:** Home, AI Tools, Run, View, Workspace, OneLake, Monitor, TechTest, Gold\_Notebook, Silver\_Notebook, student\_anal\_ytic\_ipynb..., Gold\_Warm\_out, Silver\_Warm\_out, Gold\_lease\_use.
- Top Bar:** Home, Run all, Connect, PySpark (Python), Environment, Workspace default, Data Wrangler, Copilot.
- Central Area:**
  - Explorer:** Data Items, Resources, Add data items.
  - Analytics:** Analytical queries notebook.
  - Code:** A Python code cell for querying students who have not submitted Assignment\_101.
  - Output:** The output of the code cell shows a successful JSON response with 19 rows of student data.
- Bottom Bar:** Not connected, AutoSave: On, Selected Cell 1 of 10 cells.

The screenshot shows a Databricks workspace interface. The top navigation bar includes 'Home', 'Edit', 'AI tools', 'Run', 'View', 'Comments', 'History', 'Develop', and 'Share' buttons. The left sidebar has sections for 'Home', 'Workspaces', 'OneLake', 'Monitor', 'TechTest', 'Gold, Notebooks', 'Silver, Notebooks', 'student\_mal\_ytcc\_pipeline...', 'Gold, Warehouses', 'Silverlakehouse', and 'Gold, lakehouse use'. The main area has tabs for 'PySpark (Python)', 'Environment', 'Workspace default', 'Data Wrangler', and 'Copilot'. The 'Explorer' sidebar shows 'Data items' and 'Resources' with a 'GoldLakehouse' folder containing 'SilverLakehouse'. The central workspace contains two code cells:

```
1 #Save dataFrame to gold_table
2 not_submitted_df.write.format("delta").mode("overwrite").saveAsTable("Goldlakehouse.gold_NotSubmitted")
```

[15] ✓ - Command executed in 9 sec 878 ms by Esha Fatima Shah on 10/03/23 PM, 6/03/25 PySpark (Python) ▾

```
1 # Query 2: Submission rate per course (as a %)
2
3 # Total expected submission per course
4 total_expected = silver_df.groupby("CourseID", "CourseName").agg(count("*").alias("total_expected"))
5
6 # Actual submissions per course
7 actual_submissions = silver_df.filter(col("SubmissionID").isNotNull()) \
8     .groupBy("CourseID").agg(count("*").alias("actual_submitted"))
9
10 # Join and calculate submission rate
11 submission_rate_df = total_expected.join(actual_submissions, "CourseID", "left") \
12     .withColumn("actual_submitted", F.coalesce(col("actual_submitted"), lit(0))) \
13     .withColumn("SubmissionRatePercent", round((col("actual_submitted") / col("total_expected")) * 100, 2)) \
14     .select("CourseName", "SubmissionRatePercent")
15
16 # Format JSON Output
17 submission_rate_result = {
18     "status": "success",
19     "message": "Submission rate per course",
20     "data": submission_rate_df.collect()
21 }
22
23 display(submission_rate_result)
24
```

[15] ✓ - Command executed in 940 ms by Esha Fatima Shah on 10/01/23 PM, 6/03/25 PySpark (Python) ▾

Gold\_Notebook | Saved | 16 days left

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

**Explorer**

Data items Resources

+ Add data items

Items

- GoldLakehouse
- SilverLakehouse

```
[14] 1 # Save data frame to gold table
2 overdue_df.write.format("delta").mode("overwrite").saveAsTable("GoldLakehouse.gold_SubmissionRate")
✓ - Command executed in 5 sec 314 ms by Esha Fatima Shah on 10:03:39 PM, 6/03/25
```

**Diagnostics**

```
[14] 1 # Query 3: Overdue assignment count per student
2
3 overdue_df = silver_df \
4     .filter((col("SubmissionID").isNull()) & (col("DueDate") < current_date())) \
5     .groupBy("StudentID", "Name") \
6     .agg(count("*").alias("OverdueCount"))
7
8 # Format JSON output
9 overdue_result = {
10     "status": "success",
11     "message": "Overdue assignment count per student",
12     "data": overdue_df.collect()
13 }
14
15 display(overdue_result)
✓ - Command executed in 1 sec 11 ms by Esha Fatima Shah on 10:01:33 PM, 6/03/25
```

Pyspark (Python)

Selected Cell 1 of 10 cells

Gold\_Notebook | Saved | 16 days left

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

**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='S1096', Name='Grace Watts', OverdueCount=9),
Row(StudentID='S1097', Name='Jordan Williams', OverdueCount=12),
Row(StudentID='S1098', Name='Marie Gilbert', OverdueCount=26),
Row(StudentID='S1099', Name='Scoot Cole', OverdueCount=5),
Row(StudentID='S1095', Name='Juan Calderon', OverdueCount=16),
Row(StudentID='S1096', Name='Joshua Turner', OverdueCount=16),
Row(StudentID='S1097', Name='Kaitlyn Powers', OverdueCount=11),
Row(StudentID='S1098', Name='Ashley Brennan', OverdueCount=13),
Row(StudentID='S1099', Name='Marissa Webster', OverdueCount=17),
Row(StudentID='S1099', Name='Rachel Mitchell', OverdueCount=11),
Row(StudentID='S1093', Name='Carlos Wallace', OverdueCount=17),
Row(StudentID='S1094', Name='Liam Williams', OverdueCount=17),
Row(StudentID='S1095', Name='Paige Carlson', OverdueCount=14),
Row(StudentID='S1096', Name='Vance Torres', OverdueCount=6),
Row(StudentID='S1097', Name='Austin Johnson', OverdueCount=6),
Row(StudentID='S1098', Name='Natalia Peters', OverdueCount=8),
Row(StudentID='S1099', Name='James Lewis', OverdueCount=14),
Row(StudentID='S1097', Name='Jeffrey Cox PhD', OverdueCount=7),
Row(StudentID='S1094', Name='Julie Johnson', OverdueCount=19),
Row(StudentID='S1098', Name='Kimberly Ball', OverdueCount=13),
Row(StudentID='S1099', Name='Elijah Parker', OverdueCount=12),
Row(StudentID='S1094', Name='Michael Rubin', OverdueCount=12),
Row(StudentID='S1092', Name='Valerie Williams', OverdueCount=5),
Row(StudentID='S1098', Name='Stephanie Dalton', OverdueCount=10),
Row(StudentID='S1099', Name='John Hancock', OverdueCount=8),
Row(StudentID='S1097', Name='Avery Snyder', OverdueCount=8),
Row(StudentID='S1093', Name='Brandy Wilson', OverdueCount=17),
Row(StudentID='S1099', Name='Renae Morales', OverdueCount=8),
Row(StudentID='S1094', Name='Ian Cooper', OverdueCount=23),
Row(StudentID='S1095', Name='Cameron Fisher', OverdueCount=22),
Row(StudentID='S1096', Name='Dawn Williams', OverdueCount=14),
Row(StudentID='S1090', Name='Allison Hill', OverdueCount=14),
Row(StudentID='S1091', Name='Erik Charles', OverdueCount=8),
Row(StudentID='S1093', Name='David Benson', OverdueCount=15),
Row(StudentID='S1094', Name='Liam Baker', OverdueCount=5),
Row(StudentID='S1093', Name='Tracy Higgins', OverdueCount=5),
Row(StudentID='S1091', Name='Phillip O'Brien', OverdueCount=7),
Row(StudentID='S1096', Name='Linda Dodson DVM', OverdueCount=17),
Row(StudentID='S1099', Name='Shane Henderson', OverdueCount=9),
Row(StudentID='S1097', Name='Julie Curry', OverdueCount=4),
```

11 days left

Comments History Develop Share

PySpark (Python)

Selected Cell 1 of 10 cells

Gold\_Notebook | Saved | 16 days left

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

**Explorer**

Data items Resources

+ Add data items

Items

- GoldLakehouse
- SilverLakehouse

```
[17] 1 # Save data frame to gold table
2 overdue_df.write.format("delta").mode("overwrite").saveAsTable("GoldLakehouse.gold_Overdue")
✓ - 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:

The screenshot shows two separate queries run against a "GoldLakehouse" endpoint in the OneLake service.

**Query 1: gold\_notsubmitted**

This query displays 23 rows of data from the "gold\_notsubmitted" table. The columns are StudentID, Name, and Email. The data includes various student names and their corresponding emails.

| StudentID | Name              | Email                     |
|-----------|-------------------|---------------------------|
| 1         | Anthony Everett   | ewilson@example.com       |
| 2         | Marie Gilbert     | umarshall@example.net     |
| 3         | Deborah Rodriguez | yreed@example.com         |
| 4         | Marissa Webster   | janesmith@example.org     |
| 5         | Justin Lowery     | bryantjaurie@example.net  |
| 6         | Valerie Gill      | harellkenneth@example.net |
| 7         | Michael Elliott   | stevenscott@example.com   |
| 8         | Daniel Adams      | lynchgeorge@example.net   |
| 9         | Allison Hill      | donaldgarcia@example.net  |
| 10        | Rose Spence       | davidalvarez@example.net  |
| 11        | Justin Allen      | frover@example.com        |
| 12        | Michael Martinez  | mccannangela@example.net  |
| 13        | Martin Rodriguez  | amore@example.net         |
| 14        | Joshua Smith      | jason99@example.net       |
| 15        | Page Carlson      | walkerneth@example.com    |
| 16        | Rachel Mitchell   | smoore@example.org        |
| 17        | Krista Williams   | novakasara@example.org    |
| 18        | Julie Johnson     | jasmine71@example.net     |
| 19        | Jeremy Lewis      | yelon@example.net         |
| 20        | Cameron Fisher    | lloyd78@example.net       |
| 21        | Ian Cooper        | lindsay78@example.org     |
| 22        | Daniel Baker      | nicole35@example.com      |
| 23        | Juan Calderon     | barbara10@example.net     |

**Query 2: gold\_overdue**

This query displays 90 rows of data from the "gold\_overdue" table. The columns are StudentID, Name, and OverdueCount. The data includes various student names and their overdue counts.

| StudentID | Name             | OverdueCount |
|-----------|------------------|--------------|
| 1         | Rachel Weber     | 19           |
| 2         | Grant Watts      | 10           |
| 3         | Joseph Stanley   | 12           |
| 4         | Marie Gilbert    | 27           |
| 5         | Scott Cole       | 5            |
| 6         | Juan Calderon    | 16           |
| 7         | Joshua Turner    | 16           |
| 8         | Jennifer Powers  | 11           |
| 9         | Ashley Brennan   | 13           |
| 10        | Marissa Webster  | 17           |
| 11        | Rachel Mitchell  | 11           |
| 12        | Carlos Walls     | 11           |
| 13        | Krista Williams  | 17           |
| 14        | Page Carlson     | 14           |
| 15        | Wanda Torres     | 6            |
| 16        | Austin Johnson   | 6            |
| 17        | Melissa Peterson | 9            |
| 18        | James Lewis      | 14           |
| 19        | Jeffrey Cox PhD  | 7            |
| 20        | Julie Johnson    | 19           |
| 21        | Kimberly Ball    | 13           |
| 22        | Matthew Moore    | 6            |
| 23        | Michael Rubio    | 13           |
| 24        | Valerie Williams | 5            |
| 25        | Stephanie Dalton | 10           |

The screenshot shows the Fabric interface with the 'GoldLakehouse' workspace selected. The left sidebar contains a tree view of workspaces and notebooks, with 'GoldLakehouse' expanded to show 'Tables' and 'Files'. The main area is titled 'gold\_submissionrate' and displays a table with 8 rows. The table has two columns: 'CourseName' and 'SubmissionRatePercent'. The data is as follows:

|   | CourseName                                 | SubmissionRatePercent |
|---|--|-----------------------|
| 1 | Business-focused demand-driven flexibility | 8.62                  |
| 2 | Triple-buffered cohesive function          | 11.36                 |
| 3 | Managed incremental solution               | 8.54                  |
| 4 | Organic even-keeled contingency            | 10.7                  |
| 5 | Customizable maximized capability          | 12.57                 |
| 6 | Ergonomic multi-state hierarchy            | 7.02                  |
| 7 | Automated logistical array                 | 12.9                  |
| 8 | Phased grid-enabled installation           | 8.87                  |

## 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:

The screenshot shows two views of the Microsoft Fabric Data Factory interface.

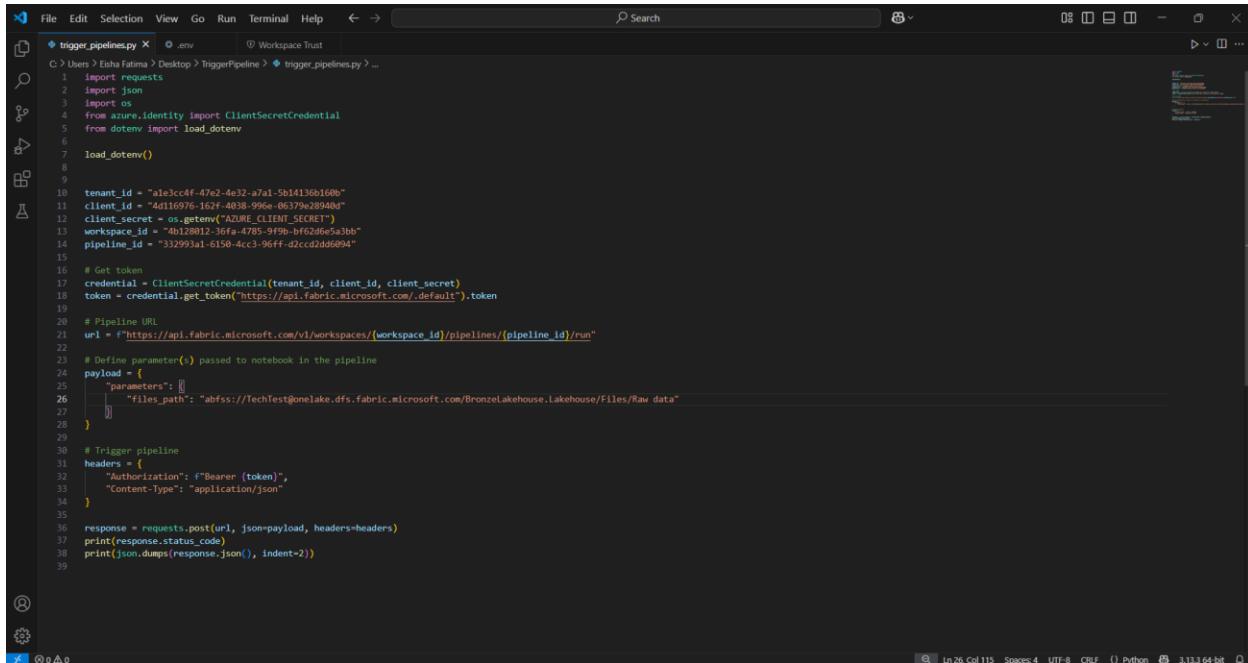
**Top View (Fabric Data Factory Pipeline Editor):**

**Bottom View (Fabric Data Factory Pipeline Run History):**

| Activity name | Activity status | Run start            | Duration | Input | Output |
|---------------|-----------------|----------------------|----------|-------|--------|
| Gold Layer    | Succeeded       | 6/9/2025, 5:54:02 AM | 1m 19s   | -0    | -0     |
| Silver Layer  | Succeeded       | 6/9/2025, 5:52:34 AM | 1m 25s   | -0    | -0     |
| Bronze Layer  | Succeeded       | 6/9/2025, 5:50:49 AM | 1m 42s   | -0    | -0     |

## Automation using Github Actions:

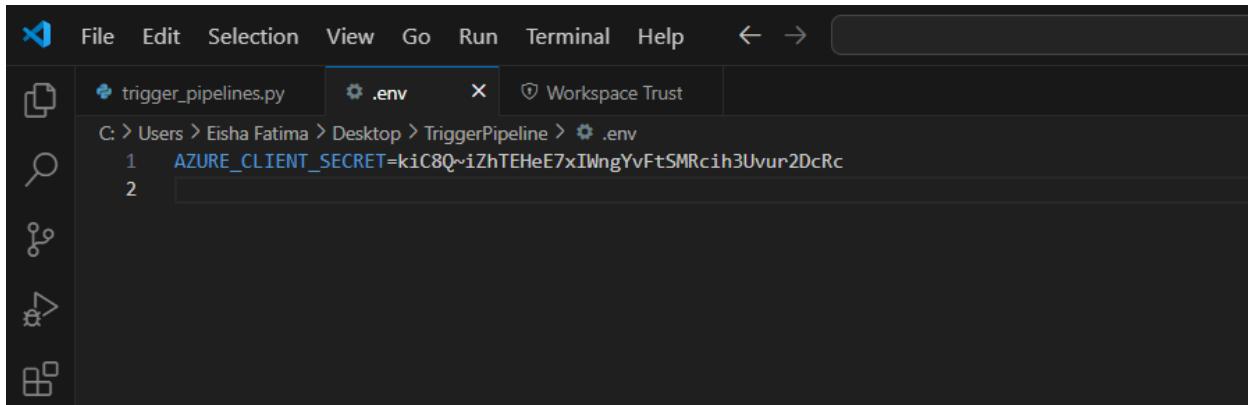
### Python script:



A screenshot of the Visual Studio Code interface. The title bar shows 'trigger\_pipelines.py' and '.env'. The code editor contains Python code for triggering a pipeline. The code imports requests, json, and ClientSecretCredential from the azure.identity module, and load\_dotenv from the dotenv module. It sets tenant\_id, client\_id, and client\_secret from environment variables. It defines a Pipeline URL and payload parameters. It triggers the pipeline using a POST request with headers and prints the response. The status bar at the bottom indicates the file has 26 lines, 115 columns, and is in Python mode.

```
File Edit Selection View Go Run Terminal Help ⏎ → Search ⌘ .env Workspace Trust
trigger_pipelines.py X .env
C:\Users\Eisha Fatima\Desktop\TriggerPipeline> trigger_pipelines.py ...
1 import requests
2 import json
3 import os
4 from azure.identity import ClientSecretCredential
5 from dotenv import load_dotenv
6
7 load_dotenv()
8
9
10 tenant_id = "ale3cc4f-47e2-4e32-a7a1-5b1a136b160b"
11 client_id = "4d116076-162f-4038-995e-06379e284040"
12 client_secret = os.getenv("AZURE_CLIENT_SECRET")
13 workspace_id = "4b128012-3efa-4785-9f9b-bfe2d6e5a3bb"
14 pipeline_id = "532995a1-615d-4cc3-96f7-d2cc2dd0b94"
15
16 # Get token
17 credential = ClientSecretCredential(tenant_id, client_id, client_secret)
18 token = credential.get_token("https://api.fabric.microsoft.com/.default").token
19
20 # Pipeline URL
21 url = f"https://api.fabric.microsoft.com/v1/workspaces/{workspace_id}/pipelines/{pipeline_id}/run"
22
23 # Define parameter(s) passed to notebook in the pipeline
24 payload = {
25     "parameters": [
26         {"files_path": "abfss://TechTest@onelake.dfs.fabric.microsoft.com/BronzeLakehouse.Lakehouse/Files/Raw data"}
27     ]
28 }
29
30 # Trigger pipeline
31 headers = {
32     "Authorization": f"Bearer {token}",
33     "Content-Type": "application/json"
34 }
35
36 response = requests.post(url, json=payload, headers=headers)
37 print(response.status_code)
38 print(json.dumps(response.json(), indent=2))
39
```

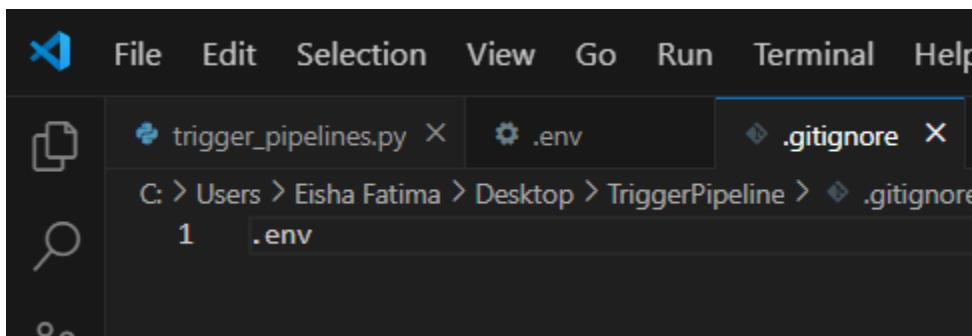
### .env file:



A screenshot of the Visual Studio Code interface. The title bar shows 'trigger\_pipelines.py', '.env', and 'Workspace Trust'. The code editor contains the environment variable AZURE\_CLIENT\_SECRET with its value set to a long string of characters. The status bar at the bottom indicates the file has 2 lines.

```
File Edit Selection View Go Run Terminal Help ⏎ →
trigger_pipelines.py .env Workspace Trust
C:\> Users > Eisha Fatima > Desktop > TriggerPipeline > .env
1 AZURE_CLIENT_SECRET=kiC8QzizhTEheE7xIWngYvFtSMRcih3Uvur2DcRc
2
```

### .gitignore file:



A screenshot of the Visual Studio Code interface. The title bar shows 'trigger\_pipelines.py', '.env', and '.gitignore'. The code editor contains the entry '.env' in the .gitignore file. The status bar at the bottom indicates the file has 1 line.

```
File Edit Selection View Go Run Terminal Help
trigger_pipelines.py .env .gitignore
C:\> Users > Eisha Fatima > Desktop > TriggerPipeline > .gitignore
1 .env
```

## Registering Entra ID app in azure:

**Microsoft Azure** Search resources, services, and docs (G+) Copilot FAST NATIONAL UNIVERSITY (N...)

Home > App registrations > Register an application

\* Name  
The user-facing display name for this application (this can be changed later).  
FabricPipelineTrigger

Supported account types  
Who can use this application or access this API?  
 Accounts in this organizational directory only (FAST National University only - Single tenant)  
 Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)  
 Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)  
 Personal Microsoft accounts only  
Help me choose...

Redirect URI (optional)  
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

By proceeding, you agree to the Microsoft Platform Policies

**Register**

**FabricPipelineTrigger | Certificates & secrets**

Overview Quickstart Integration assistant Diagnose and solve problems Manage Branding & properties Authentication Certificates & secrets

Got feedback? Got a second to give us some feedback? →

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) Client secrets (1) Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

New client secret

| Description                          | Expires   | Value                                | Secret ID                            |
|--------------------------------------|-----------|--------------------------------------|--------------------------------------|
| Password uploaded on Sun Jun 08 2025 | 12/5/2025 | kiC8Q~IzHTEHeE7xIWngYvFtSMRcih3Uv... | 1a8e01f1-0d85-45db-ad2d-99a1f0771ddd |

Add or remove favorites by pressing Ctrl+Shift+F

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

k200383@nu.edu.pk

FAST NATIONAL UNIVERSITY (NU)

FabricPipelineTrigger

Home > App registrations >

FabricPipelineTrigger

Search

Delete Endpoints Preview features

Overview

Quickstart

Integration assistant

Diagnose and solve problems

Manage

Support + Troubleshooting

Essentials

|                             |  |
|-----------------------------|--|
| Display name                | : FabricPipelineTrigger                |
| Application (client) ID     | : 4d116976-162f-4038-996e-06379e28940d |
| Object ID                   | : b0010b30-89d-4200-a7f8-74cec5142931  |
| Directory (tenant) ID       | : a1e3cc4f-47e2-4e32-a7a1-5b14136b160b |
| Supported account types     | : My organization only                 |
| Client credentials          | : Add a certificate or secret          |
| Redirect URIs               | : Add a Redirect URI                   |
| Application ID URI          | : Add an Application ID URI            |
| Managed application in I... | : FabricPipelineTrigger                |

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? [Learn more](#)

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)

Get Started Documentation

Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. [Learn more](#)

Add or remove favorites by pressing Ctrl+Shift+F

## Pushing files to GitHub repository via git:

```
MINGW64:/c/Users/Eisha Fatima/Desktop/TriggerPipeline
Unpacking objects: 100% (3/3), 879 bytes | 125.00 KiB/s, done.
From https://github.com/k200383/StudentAnalytics
 * branch            main      -> FETCH_HEAD
 * [new branch]      main      -> origin/main
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

Eisha Fatima@DESKTOP-RIRHOK6 MINGW64 ~/Desktop/TriggerPipeline (main)
$ git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.16 KiB | 1.16 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/k200383/StudentAnalytics
 51baca3..ad5ca88 main -> main
branch 'main' set up to track 'origin/main'.

Eisha Fatima@DESKTOP-RIRHOK6 MINGW64 ~/Desktop/TriggerPipeline (main)
```

The screenshot shows a GitHub repository page for 'StudentAnalytics'. The repository has 1 branch and 0 tags. It contains three commits from user 'k200383'. The README file is visible. The repository has 0 stars, 0 forks, and 0 releases. It uses Python as its primary language.

**Code** | **Issues** | **Pull requests** | **Actions** | **Projects** | **Wiki** | **Security** | **Insights** | **Settings**

**StudentAnalytics** Public

main 1 Branch 0 Tags

k200383 Merge branch 'main' of https://github.com/k200383/StudentAnalytics ad5ca88 · now 3 Commits

.gitignore initial 2 minutes ago

README.md Initial commit 6 minutes ago

trigger\_pipelines.py initial 2 minutes ago

**README**

## StudentAnalytics

About

No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Languages

Python 100.0%

## Adding Github secrets:

The screenshot shows the GitHub Secrets settings page. Under 'Environment secrets', it says 'This environment has no secrets.' and has a 'Manage environment secrets' button. Under 'Repository secrets', there are five secrets listed:

| Name                 | Last updated | Action      |
|----------------------|--------------|-------------|
| FABRIC_CLIENT_ID     | 1 minute ago | edit delete |
| FABRIC_CLIENT_SECRET | now          | edit delete |
| FABRIC_PIPELINE_ID   | now          | edit delete |
| FABRIC_TENANT_ID     | 1 minute ago | edit delete |
| FABRIC_WORKSPACE_ID  | now          | edit delete |

Moderation options

about variables.

Code and automation

Branches

Tags

Rules

Actions

Models

Webhooks

Environments

Codespaces

Pages

Secrets and variables

Actions

Codespaces

Dependabot

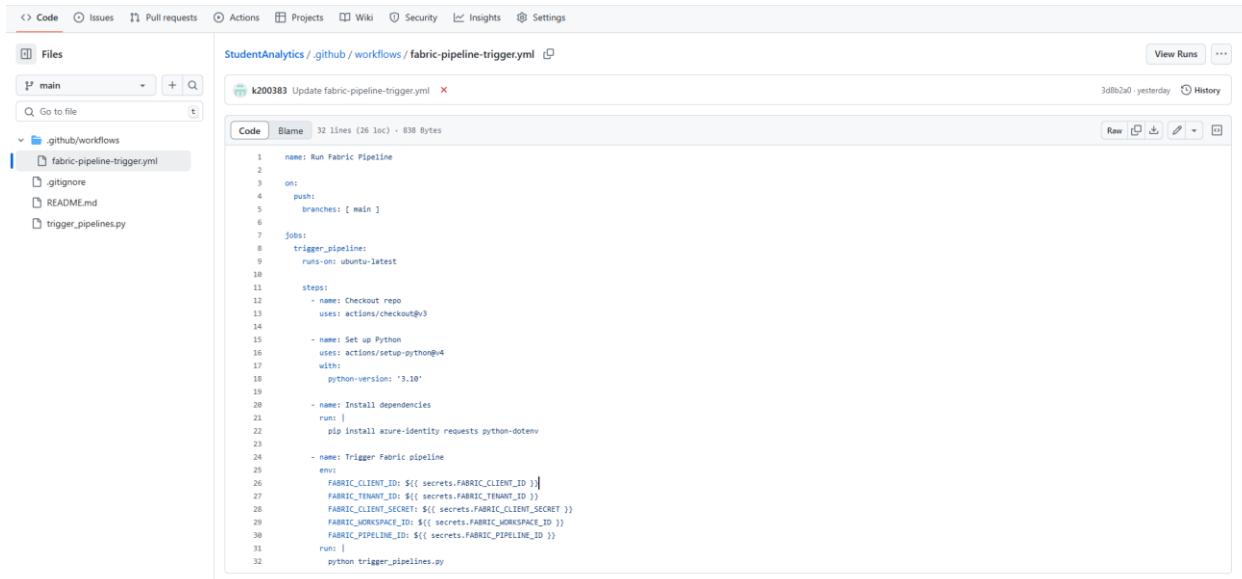
Integrations

GitHub Apps

Email notifications

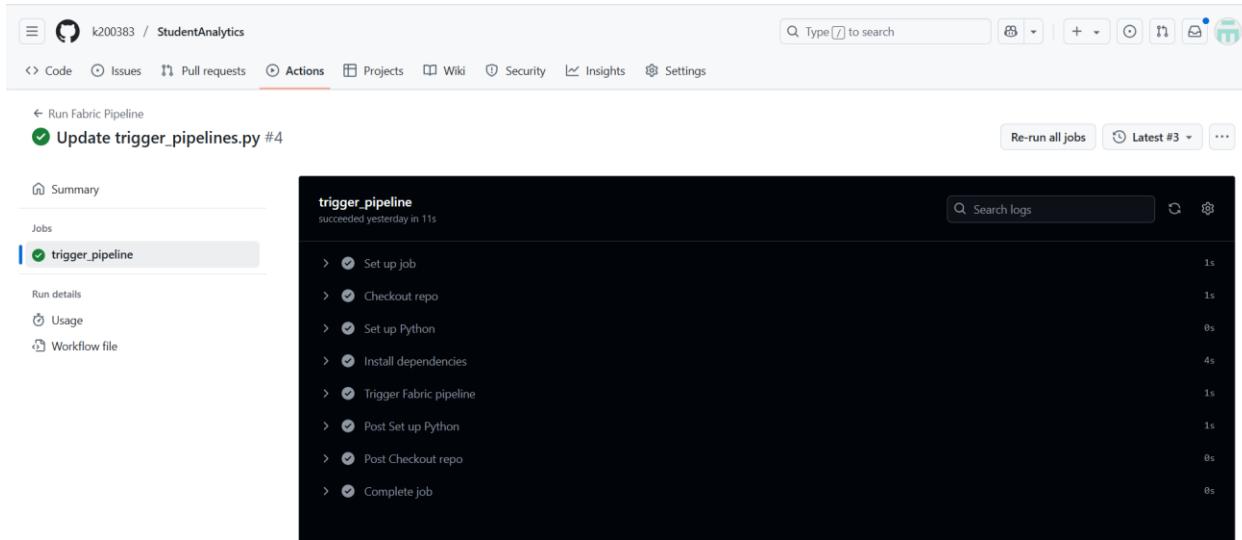
New repository secret

## YML code for Github actions workflow:



The screenshot shows the GitHub Actions workflow configuration file, `fabric-pipeline-trigger.yml`. The file defines a workflow named "Run Fabric Pipeline" that triggers on pushes to the main branch. It consists of several steps: checkout the repository, set up Python 3.10, install dependencies using pip, and finally trigger the Fabric pipeline using a Python script. Secrets for client ID, secret, workspace ID, and pipeline ID are used in the trigger command.

```
name: Run Fabric Pipeline
on:
  push:
    branches: [ main ]
jobs:
  trigger_pipeline:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repo
        uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@4
        with:
          python-version: '3.10'
      - name: Install dependencies
        run:
          pip install azure-identity requests python-dotenv
      - name: Trigger Fabric pipeline
        env:
          FABRIC_CLIENT_ID: ${{ secrets.FABRIC_CLIENT_ID }}
          FABRIC_CLIENT_SECRET: ${{ secrets.FABRIC_CLIENT_SECRET }}
          FABRIC_WORKSPACE_ID: ${{ secrets.FABRIC_WORKSPACE_ID }}
          FABRIC_PIPELINE_ID: ${{ secrets.FABRIC_PIPELINE_ID }}
        run:
          python trigger_pipelines.py
```



The screenshot shows the logs for the "trigger\_pipeline" job. The job succeeded yesterday in 11s. The log details the execution of the workflow steps: Set up job, Checkout repo, Set up Python, Install dependencies, and Trigger Fabric pipeline. Each step is marked with a green checkmark indicating success.

trigger\_pipeline succeeded yesterday in 11s

| Step                    | Description | Time |
|-------------------------|-------------|------|
| Set up job              | > ✓         | 1s   |
| Checkout repo           | > ✓         | 1s   |
| Set up Python           | > ✓         | 0s   |
| Install dependencies    | > ✓         | 4s   |
| Trigger Fabric pipeline | > ✓         | 1s   |
| Post Set up Python      | > ✓         | 1s   |
| Post Checkout repo      | > ✓         | 0s   |
| Complete job            | > ✓         | 0s   |

**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):**

```
Administrator: Windows PowerShell
program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:4 char:1
+ AccessRight Contributor
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (AccessRight:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\WINDOWS\system32> Add-PowerBIWorkspaceUser -Id "4b128012-36fa-4785-9f9b-bf62d6e5a3bb" PrincipalType App Identifier
"4d116976-162f-4038-996e-06379e28940d" AccessRight Contributor
Add-PowerBIWorkspaceUser : A positional parameter cannot be found that accepts argument 'PrincipalType'.
At line:1 char:1
+ Add-PowerBIWorkspaceUser -Id "4b128012-36fa-4785-9f9b-bf62d6e5a3bb" P ...
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (:) [Add-PowerBIWorkspaceUser], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerBI.Commands.Workspaces.AddPowerBIWorkspaceUse
r

PS C:\WINDOWS\system32> Add-PowerBIWorkspaceUser -Id "4b128012-36fa-4785-9f9b-bf62d6e5a3bb" -PrincipalType App -Identifier
"4d116976-162f-4038-996e-06379e28940d" -AccessRight Contributor
Add-PowerBIWorkspaceUser : Operation returned an invalid status code 'Forbidden'
At line:1 char:1
+ Add-PowerBIWorkspaceUser -Id "4b128012-36fa-4785-9f9b-bf62d6e5a3bb" - ...
+ ~~~~~
+ CategoryInfo          : WriteError: (Microsoft.Power...BIWorkspaceUser:AddPowerBIWorkspaceUser) [Add-PowerBIWork
spaceUser], HttpClientException
+ FullyQualifiedErrorId : Operation returned an invalid status code 'Forbidden',Microsoft.PowerBI.Commands.Workspa
ces.AddPowerBIWorkspaceUser

PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32>
```

):

FabricPipelineTrigger | API permissions

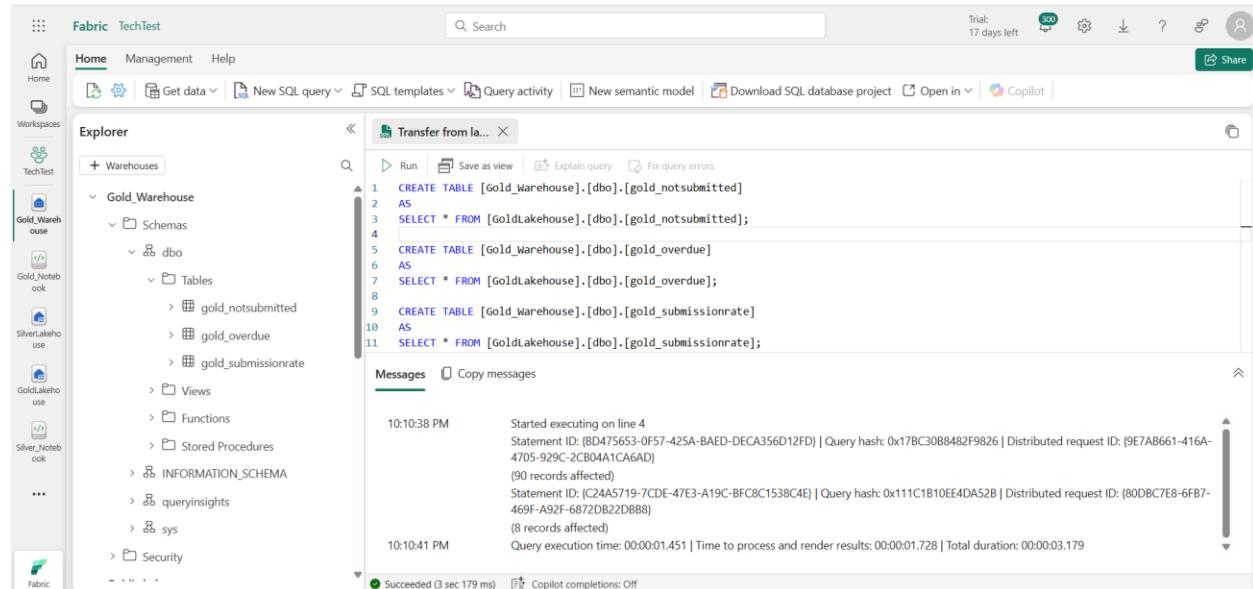
Configured permissions

| API / Permissions name     | Type      | Description  | Admin consent req... | Status |
|----------------------------|-----------|--|----------------------|--------|
| User.Read                  | Delegated | Sign in and read user profile                                  | No                   | ***    |
| DataPipeline.Execute.All   | Delegated | Make API calls that require execute permissions all data pi... | No                   | ***    |
| DataPipeline.Read.All      | Delegated | Make API calls that require read permissions on all data pi... | No                   | ***    |
| DataPipeline.ReadWrite.All | Delegated | Make API calls that require read and write permissions on ...  | No                   | ***    |
| DataPipeline.Reshare.All   | Delegated | Make API calls that require reshare permissions on all data... | No                   | ***    |

## Data governance

### Row-level security:

### Transfer data from lakehouse to warehouse:



The screenshot shows the Fabric interface with the 'TechTest' workspace selected. In the top navigation bar, 'Fabric' and 'TechTest' are visible along with 'Management' and 'Help'. Below the navigation bar is a toolbar with various icons for 'Get data', 'New SQL query', 'SQL templates', 'Query activity', 'New semantic model', 'Download SQL database project', 'Open in', 'Copilot', and 'Share'. The left sidebar is titled 'Workspaces' and lists several databases: Gold\_Warehouse, Gold\_Notebook, SilverLakehouse, GoldLakehouse, and Silver\_Notebook. The 'Gold\_Warehouse' database is currently selected. The main area is titled 'Transfer from la...' and contains a code editor with the following SQL script:

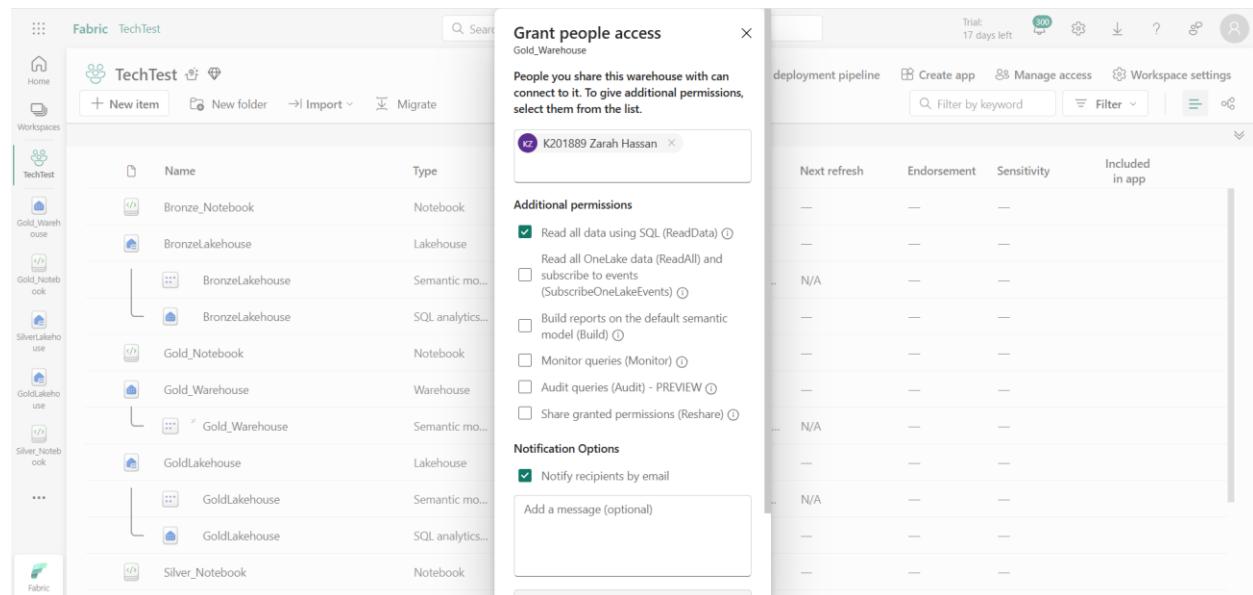
```
1 CREATE TABLE [Gold_Warehouse].[dbo].[gold_notsubmitted]
2 AS
3 SELECT * FROM [GoldLakehouse].[dbo].[gold_notsubmitted];
4
5 CREATE TABLE [Gold_Warehouse].[dbo].[gold_overdue]
6 AS
7 SELECT * FROM [GoldLakehouse].[dbo].[gold_overdue];
8
9 CREATE TABLE [Gold_Warehouse].[dbo].[gold_submissionrate]
10 AS
11 SELECT * FROM [GoldLakehouse].[dbo].[gold_submissionrate];
```

Below the code editor, there is a 'Messages' section showing two log entries:

- 10:10:38 PM Started executing on line 4 Statement ID: (B0475653-0F57-425A-BAED-DECA356D12FD) | Query hash: 0x17BC30B8482F9826 | Distributed request ID: (9E7AB661-416A-4705-929C-2CB04A1CA6AD)  
(90 records affected)
- 10:10:41 PM Statement ID: (C24A5719-7CDE-47E3-A19C-BFC8C1538C4E) | Query hash: 0x111C1B10EE4DA52B | Distributed request ID: (80DBC7E8-6FB7-469F-A92F-6872DB22DB88)  
(8 records affected)

The status bar at the bottom indicates 'Succeeded (3 sec 179 ms)' and 'Copilot completions: Off'.

### Giving only read access of warehouse to user:



The screenshot shows the Fabric interface with the 'TechTest' workspace selected. In the top navigation bar, 'Fabric' and 'TechTest' are visible along with 'Management' and 'Help'. Below the navigation bar is a toolbar with various icons for 'Create app', 'Manage access', and 'Workspace settings'. The left sidebar is titled 'Workspaces' and lists several databases: Gold\_Warehouse, Gold\_Notebook, SilverLakehouse, GoldLakehouse, and Silver\_Notebook. The 'Gold\_Warehouse' database is currently selected. A modal window titled 'Grant people access' is open over the workspace list, showing the user 'K201889 Zarah Hassan' selected. The modal has sections for 'Additional permissions' and 'Notification Options'. Under 'Additional permissions', the checkbox 'Read all data using SQL (ReadData)' is checked. Under 'Notification Options', the checkbox 'Notify recipients by email' is checked. The right side of the screen shows a table for the 'deployment pipeline' with columns: 'Next refresh', 'Endorsement', 'Sensitivity', and 'Included in app'. Most values in the table are 'N/A'.

The screenshot shows the Microsoft Fabric interface. On the left, there's a sidebar with icons for Home, Workspaces, TechTest, and various notebooks. The main area is titled "Fabric" and "Gold\_Warehouse". It has a search bar and a "Direct access" section. Under "People and groups with access", two users are listed: Eisha Fatima Shah (Workspace Admin) and K201889 Zarah Hassan (Read permission). There are also "Filter" and "... more" buttons.

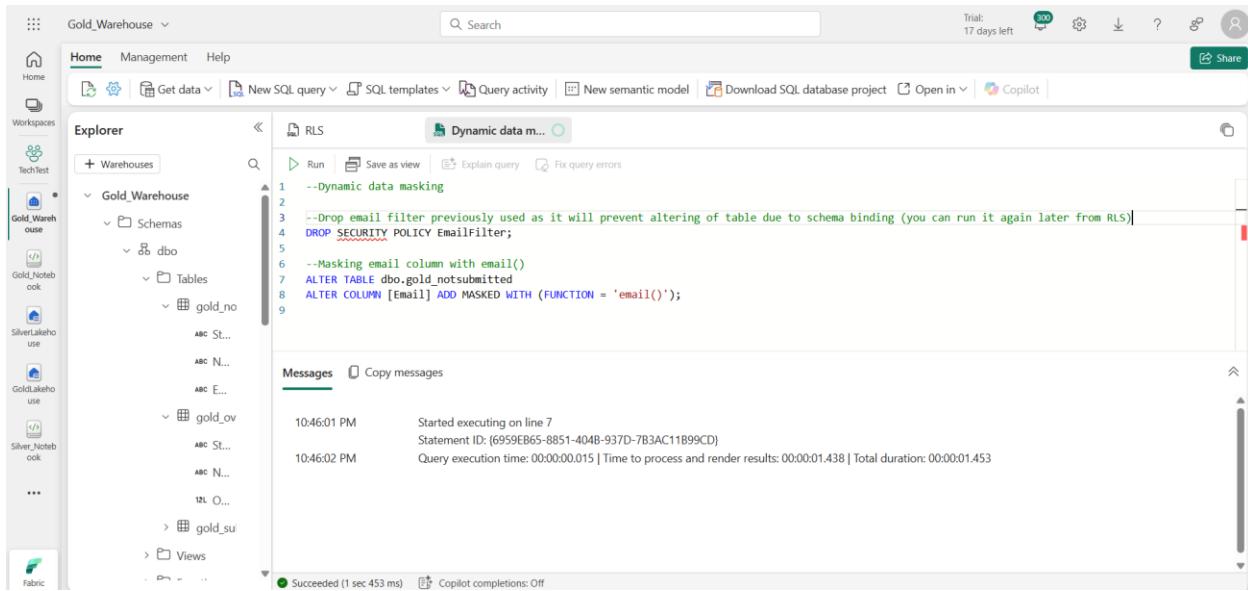
Make sure user is having only read permission to warehouse so they cannot see the data of all the warehouse tables unless access is granted to them. User can see only their email's associated information (row-level security):

The screenshot shows the Microsoft Fabric Explorer interface. The sidebar includes Home, Reporting, Management, Help, and various workspace icons. The main area shows the "Explorer" pane with "Gold\_Warehouse" selected, and the "Data preview - gold\_notsubmitted" tab. The preview shows a single row of data:

|   | ABC StudentID | Name         | Email             |
|---|---------------|--------------|-------------------|
| 1 | S1111         | Zarah Hassan | k201889@nu.edu.pk |

At the bottom, it says "Showing 1000 rows" and "Succeeded (1 sec 343 ms)".

## Dynamic data masking:



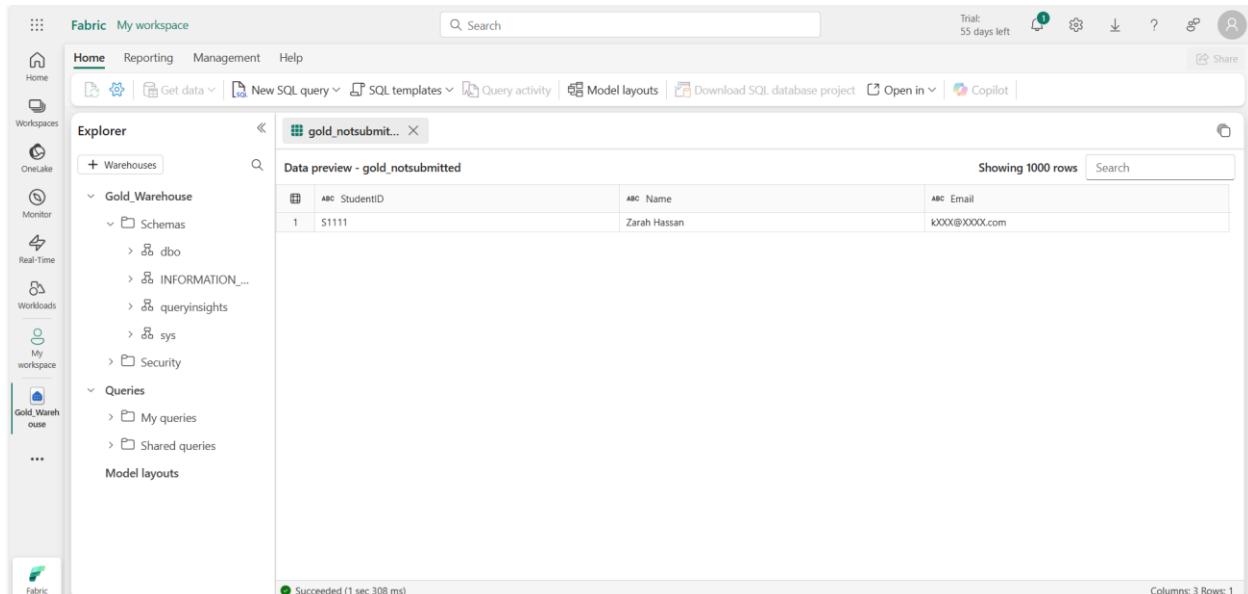
The screenshot shows the Fabric interface with the 'Gold\_Warehouse' workspace selected. In the center, there's a code editor titled 'Dynamic data m...'. The code is as follows:

```
1 --Dynamic data masking
2
3 --Drop email filter previously used as it will prevent altering of table due to schema binding (you can run it again later from RLS)
4 DROP SECURITY POLICY EmailFilter;
5
6 --Masking email column with email()
7 ALTER TABLE dbo.gold_notsubmitted
8 ALTER COLUMN [Email] ADD MASKED WITH (FUNCTION = 'email()');
9
```

Below the code editor is a 'Messages' section showing logs of the execution:

- 10:46:01 PM Started executing on line 7
- Statement ID: (6959EB65-8851-404B-937D-7B3AC11B99CD)
- 10:46:02 PM Query execution time: 00:00:00.015 | Time to process and render results: 00:00:01.438 | Total duration: 00:00:01.453

At the bottom, a status bar indicates 'Succeeded (1 sec 453 ms)' and 'Copilot completions: Off'.



The screenshot shows the Fabric interface with the 'Fabric My workspace' workspace selected. In the center, there's a 'Data preview - gold\_notsubmitted' section. The table has three columns: StudentID, Name, and Email. There is one row of data:

|   | ABC StudentID | ABC Name     | ABC Email     |
|---|---------------|--------------|---------------|
| 1 | S1111         | Zarah Hassan | 1000@XXXX.com |

At the bottom, a status bar indicates 'Succeeded (1 sec 308 ms)' and 'Columns: 3 Rows: 1'.

## Column-level security and object-level security:

The screenshot displays two separate sessions in the Azure Data Studio interface.

**Gold\_Warehouse Session:**

- Explorer:** Shows the warehouse structure. Under the `Gold_Warehouse` schema, there are tables: `gold_notsubmitted`, `gold_overdue`, and `gold_submissionrate`. The `gold_overdue` table has columns: `StudentID`, `Name`, and `Email`.
- Query Editor:** A query titled "CLS + OLS" is running:

```
1 --Column level security + object level security
2 GRANT SELECT ([StudentID], [OverdueCount])
3 ON dbo.gold_overdue
4 TO "k201889@nu.edu.pk" --will grant access to only 2 columns out of 3
5
6 GRANT SELECT
7 ON dbo.gold_notsubmitted
8 TO "k201889@nu.edu.pk" --will grant access to this table to user
```
- Messages:** Shows the execution status: "Succeeded (0 sec 802 ms)".

**Fabric Session:**

- Explorer:** Shows the warehouse structure. Under the `Gold_Warehouse` schema, there are tables: `gold_overdue` and `gold_notsubmitted`. The `gold_overdue` table has columns: `StudentID`, `Name`, and `Email`.
- Query Editor:** A query titled "Read gold\_over..." is running:

```
1 SELECT StudentID, OverdueCount FROM dbo.gold_overdue;
```
- Results:** Displays the results of the query:

| StudentID | OverdueCount |
|-----------|--------------|
| S1095     | 19           |
| S1038     | 10           |
| S1076     | 12           |
| S1078     | 27           |
| S1061     | 5            |
| S1005     | 16           |
| S1036     | 16           |
| S1020     | 11           |

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

Fabric My workspace

Home Reporting Management Help

Get data New SQL query SQL templates Query activity Model layouts Download SQL database project Open in Copilot

Search Trial: 55 days left

Share

Workspaces OneLake Home Monitor Real-Time Workloads My workspace Gold\_Warehouse

Explorer

+ Warehouses

Gold\_Warehouse

- Schemas
- Tables
  - gold\_no
  - gold\_ov
- Views
- Functions
- Stored Pr...
- INFORMATIO...
- queryinsights
- sys
- Security

Queries

MV queries

gold\_overdue | gold\_notsubmitt... | Read gold\_overdue...

Run Save as view Explain query Fix query errors

Preview Copilot uses AI. Mistakes can happen. Verify code suggestions before running them. [Review terms](#)

```
1 SELECT * FROM dbo.gold_overdue;
```

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'.  
Query execution time: 00:00:02.622 | Time to process and render results: 00:00:01.400 | Total duration: 00:00:04.022

11:07:20 PM

Completed with errors (4 sec 22 ms) Copilot completions: Off

This screenshot shows the Microsoft Fabric SQL interface. The top navigation bar includes 'Fabric My workspace', 'Home', 'Reporting', 'Management', 'Help', and various connectivity options like 'Get data', 'New SQL query', 'SQL templates', 'Query activity', 'Model layouts', 'Download SQL database project', 'Open in', and 'Copilot'. A trial period of 55 days left is displayed. The left sidebar contains links for 'OneLake', 'Home', 'Monitor', 'Real-Time', 'Workloads', 'My workspace', and a specific 'Gold\_Warehouse' entry. The main area has an 'Explorer' sidebar with sections for '+ Warehouses', 'Gold\_Warehouse' (schemas, tables, views, functions, stored procedures, information, query insights, sys, security), and 'Queries' (MV queries). A preview window for 'gold\_overdue' shows a single line of SQL: 'SELECT \* FROM dbo.gold\_overdue;'. Below it, a 'Messages' pane displays log entries from 11:07:16 PM to 11:07:20 PM, detailing the execution of the query and a permission denial error for the 'Name' column in the 'gold\_overdue' table. The bottom status bar indicates the query completed with errors in 4 seconds and 22 milliseconds, with copilot completions off.