```
!pip install psycopg2-binary pandas sqlalchemy boto3
```

```
Collecting psycopg2-binary
  Downloading psycopg2_binary-2.9.10-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.9 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.11/dist-packages (2.0.40)
Collecting boto3
  Downloading boto3-1.37.26-py3-none-any.whl.metadata (6.7 kB)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.11/dist-packages (from sqlalchemy) (3.1.1)
Requirement already satisfied: typing-extensions>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from sqlalchemy) (4.
Collecting botocore<1.38.0,>=1.37.26 (from boto3)
  Downloading botocore-1.37.26-py3-none-any.whl.metadata (5.7 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
Collecting s3transfer<0.12.0,>=0.11.0 (from boto3)
  Downloading s3transfer-0.11.4-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: urllib3!=2.2.0,<3,>=1.25.4 in /usr/local/lib/python3.11/dist-packages (from botocore<1.38
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas)
Downloading psycopg2_binary-2.9.10-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 26.6 MB/s eta 0:00:00
Downloading boto3-1.37.26-py3-none-any.whl (139 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.6/139.6 kB 6.9 MB/s eta 0:00:00
Downloading botocore-1.37.26-py3-none-any.whl (13.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.5/13.5 MB 27.5 MB/s eta 0:00:00
Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Downloading s3transfer-0.11.4-py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.4/84.4 kB 5.6 MB/s eta 0:00:00
Installing collected packages: psycopg2-binary, jmespath, botocore, s3transfer, boto3
Successfully installed boto3-1.37.26 botocore-1.37.26 jmespath-1.0.1 psycopg2-binary-2.9.10 s3transfer-0.11.4
```

```
# Import required libraries
import pandas as pd
from sqlalchemy import create_engine, text
import boto3
from botocore.exceptions import ClientError
from datetime import datetime
import os
import shutil
from pathlib import Path
import logging

# AWS Configuration
AWS_ACCESS_KEY = 'AKIAQH72IQQP77ZGNJFB'
AWS_SECRET_KEY = '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'
AWS_REGION = 'ap-south-1'
BUCKET_NAME = 'testempdoc'

print("Libraries imported and AWS configured successfully!")
```

```
Libraries imported and AWS configured successfully!
```

```
def create_test_files():
    try:
        print("Creating test files...")

        # Create test directory
        test_dir = Path("/Users/espinshalo/Downloads/FDMS/test_documents")
        test_dir.mkdir(parents=True, exist_ok=True)

        # Define test files and their content
        test_files = {
            "Book.xlsx": "Sample Excel content for employee records\nEmployee ID: 001\nDepartment: HR",
            "details.pdf": "Sample PDF content for employee details\nEmployee: John Doe\nPosition: Manager",
            "Dummy.docx": "Sample Word document content\nProject: FDMS\nStatus: Active",
            "payslip.pdf": "Sample payslip content\nEmployee: Jane Smith\nMonth: April 2024",
            "position.pdf": "Sample position document content\nRole: Senior Developer\nDepartment: Engineering",
            "salary.pdf": "Sample salary document content\nEmployee: Bob Johnson\nYear: 2024",
            "first_source_jpg.png": "Sample image content for first source\nType: PNG\nSize: 9.3KB",
            "firstsource_logo_jpeg.jpeg": "Sample image content for Firstsource Logo\nType: JPEG\nSize: 17KB",
            "firstsource_logo.jpg": "Sample image content for Firstsource Logo\nType: JPG\nSize: 17KB"
        }

        created_files = []
        # Create each test file
        for filename, content in test_files.items():
            file_path = test_dir / filename
            with open(file_path, 'w') as f:
                f.write(f"{content}\nCreated at: {datetime.now()}")
```

```
                created_files.append(str(file_path))
                print(f"Created: {file_path}")

            print(f"\nCreated {len(created_files)} files successfully!")
            return created_files

        except Exception as e:
            print(f"Error creating test files: {str(e)}")
            return []

    # Create test files
    test_files = create_test_files()
```

```
Creating test files...
Created: /Users/espinshalo/Downloads/FDMS/test_documents/Book.xlsx
Created: /Users/espinshalo/Downloads/FDMS/test_documents/details.pdf
Created: /Users/espinshalo/Downloads/FDMS/test_documents/Dummy.docx
Created: /Users/espinshalo/Downloads/FDMS/test_documents/payslip.pdf
Created: /Users/espinshalo/Downloads/FDMS/test_documents/position.pdf
Created: /Users/espinshalo/Downloads/FDMS/test_documents/salary.pdf
Created: /Users/espinshalo/Downloads/FDMS/test_documents/first_source_jpg.png
Created: /Users/espinshalo/Downloads/FDMS/test_documents/firstsource_logo_jpeg.jpeg
Created: /Users/espinshalo/Downloads/FDMS/test_documents/firstsource_logo.jpg

Created 9 files successfully!
```

```
def create_old_database():
    try:
        print("Creating old database with test file paths...")

        # Create SQLite database for old system
        old_db = create_engine('sqlite:///old_db.db')

        # Create tables
        with old_db.connect() as conn:
            # Drop existing tables
            conn.execute(text("DROP TABLE IF EXISTS employee_documents"))
            conn.execute(text("DROP TABLE IF EXISTS employees"))
            conn.execute(text("DROP TABLE IF EXISTS departments"))

            # Create departments table
            conn.execute(text("""
            CREATE TABLE departments (
                department_id INTEGER PRIMARY KEY AUTOINCREMENT,
                department_name TEXT NOT NULL
            )
            """))

            # Create employees table
            conn.execute(text("""
            CREATE TABLE employees (
                employee_id INTEGER PRIMARY KEY AUTOINCREMENT,
                first_name TEXT NOT NULL,
                last_name TEXT NOT NULL,
                email TEXT NOT NULL UNIQUE,
                department_id INTEGER,
                status TEXT
            )
            """))

            # Create employee_documents table
            conn.execute(text("""
            CREATE TABLE employee_documents (
                document_id INTEGER PRIMARY KEY AUTOINCREMENT,
                employee_id INTEGER,
                document_type TEXT NOT NULL,
                file_path TEXT NOT NULL,
                upload_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                document_number TEXT,
                FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
            )
            """))

            # Insert departments
            conn.execute(text("""
            INSERT INTO departments (department_name) VALUES
                ('HR'),
                ('Finance'),
                ('Operations')
            """))

            # Insert employees
            conn.execute(text("""
```

```
            INSERT INTO employees
            (first_name, last_name, email, department_id, status) VALUES
                ('John', 'Doe', 'john.doe@company.com', 1, 'ACTIVE'),
                ('Jane', 'Smith', 'jane.smith@company.com', 2, 'ACTIVE'),
                ('Bob', 'Johnson', 'bob.johnson@company.com', 3, 'ACTIVE')
            """))

            # Insert documents with test file paths
            test_dir = Path("/Users/espinshalo/Downloads/FDMS/test_documents")
            documents = [
                (1, 'EXCEL', str(test_dir / "Book.xlsx"), 'DOC001'),
                (1, 'PDF', str(test_dir / "details.pdf"), 'DOC002'),
                (1, 'IMAGE', str(test_dir / "first_source_jpg.png"), 'IMG001'),
                (2, 'WORD', str(test_dir / "Dummy.docx"), 'DOC003'),
                (2, 'PAYSLIP', str(test_dir / "payslip.pdf"), 'DOC004'),
                (2, 'IMAGE', str(test_dir / "firstsource_logo_jpeg.jpeg"), 'IMG002'),
                (3, 'POSITION', str(test_dir / "position.pdf"), 'DOC005'),
                (3, 'SALARY', str(test_dir / "salary.pdf"), 'DOC006'),
                (3, 'IMAGE', str(test_dir / "firstsource_logo.jpg"), 'IMG003')
            ]

            for emp_id, doc_type, file_path, doc_num in documents:
                conn.execute(text("""
                INSERT INTO employee_documents
                (employee_id, document_type, file_path, document_number)
                VALUES (:emp_id, :doc_type, :file_path, :doc_num)
                """), {
                    'emp_id': emp_id,
                    'doc_type': doc_type,
                    'file_path': file_path,
                    'doc_num': doc_num
                })

            conn.commit()

        print("Old database created successfully!")

        # Verify the data
        with old_db.connect() as conn:
            print("\nVerifying data in old database:")
            documents = pd.read_sql("""
                SELECT
                    ed.document_id,
                    ed.document_type,
                    ed.file_path,
                    e.first_name,
                    e.last_name,
                    d.department_name
                FROM employee_documents ed
                JOIN employees e ON ed.employee_id = e.employee_id
                JOIN departments d ON e.department_id = d.department_id
            """, conn)

            print("\nDocument Mappings:")
            print(documents)

            # Verify file existence
            print("\nVerifying file existence:")
            for _, row in documents.iterrows():
                file_exists = os.path.exists(row['file_path'])
                print(f"File: {row['file_path']} - {'Exists' if file_exists else 'Not Found'}")

        return True

    except Exception as e:
        print(f"Error creating old database: {str(e)}")
        return False

# Create old database
create_old_database()
```

```
Creating old database with test file paths...
Old database created successfully!

Verifying data in old database:

Document Mappings:
   document_id document_type  \
0            1         EXCEL
1            2           PDF
2            3         IMAGE
3            4          WORD
4            5       PAYSLIP
```

```
5            6         IMAGE
6            7      POSITION
7            8        SALARY
8            9         IMAGE

                                          file_path first_name last_name  \
0  /Users/espinshalo/Downloads/FDMS/test_document...       John       Doe
1  /Users/espinshalo/Downloads/FDMS/test_document...       John       Doe
2  /Users/espinshalo/Downloads/FDMS/test_document...       John       Doe
3  /Users/espinshalo/Downloads/FDMS/test_document...       Jane     Smith
4  /Users/espinshalo/Downloads/FDMS/test_document...       Jane     Smith
5  /Users/espinshalo/Downloads/FDMS/test_document...       Jane     Smith
6  /Users/espinshalo/Downloads/FDMS/test_document...        Bob   Johnson
7  /Users/espinshalo/Downloads/FDMS/test_document...        Bob   Johnson
8  /Users/espinshalo/Downloads/FDMS/test_document...        Bob   Johnson

  department_name
0              HR
1              HR
2              HR
3         Finance
4         Finance
5         Finance
6      Operations
7      Operations
8      Operations

Verifying file existence:
File: /Users/espinshalo/Downloads/FDMS/test_documents/Book.xlsx - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/details.pdf - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/first_source_jpg.png - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/Dummy.docx - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/payslip.pdf - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/firstsource_logo_jpeg.jpeg - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/position.pdf - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/salary.pdf - Exists
File: /Users/espinshalo/Downloads/FDMS/test_documents/firstsource_logo.jpg - Exists
True
```

```
def create_new_database():
    try:
        print("Creating new database schema...")

        # Create SQLite database for new system
        new_db = create_engine('sqlite:///new_db.db')

        # Create document_metadata table
        with new_db.connect() as conn:
            # Drop existing table if it exists
            conn.execute(text("DROP TABLE IF EXISTS document_metadata"))

            # Create new table
            conn.execute(text("""
            CREATE TABLE document_metadata (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                document_id TEXT NOT NULL UNIQUE,
                employee_id INTEGER NOT NULL,
                first_name TEXT NOT NULL,
                last_name TEXT NOT NULL,
                email TEXT NOT NULL,
                department_name TEXT NOT NULL,
                document_type TEXT NOT NULL,
                document_category TEXT NOT NULL,
                document_number TEXT,
                original_file_path TEXT NOT NULL,
                s3_file_path TEXT,
                upload_date TIMESTAMP NOT NULL,
                processed_date TIMESTAMP,
                status TEXT NOT NULL,
                is_active BOOLEAN DEFAULT 1
            )
            """))

            conn.commit()

        print("New database schema created successfully!")
        return True

    except Exception as e:
        print(f"Error creating new database: {str(e)}")
        return False

# Create new database schema
create_new_database()
```

```
Creating new database schema...
New database schema created successfully!
True
```

```
def migrate_data_to_new_db():
    try:
        print("Starting data migration from old to new database...")

        # Connect to databases
        old_db = create_engine('sqlite:///old_db.db')
        new_db = create_engine('sqlite:///new_db.db')

        # Extract data from old database
        query = """
        SELECT
            ed.document_id as old_document_id,
            ed.employee_id,
            ed.document_type,
            ed.file_path,
            ed.document_number,
            ed.upload_date,
            e.first_name,
            e.last_name,
            e.email,
            e.status,
            d.department_name
        FROM employee_documents ed
        JOIN employees e ON ed.employee_id = e.employee_id
        JOIN departments d ON e.department_id = d.department_id
        """

        df = pd.read_sql(query, old_db)
        print(f"Found {len(df)} records to migrate")

        # Create a single connection for all operations
        with new_db.connect() as conn:
            # Start transaction
            trans = conn.begin()
            try:
                for idx, row in df.iterrows():
                    # Create unique document ID
                    document_id = f"DOC_{row['employee_id']}_{row['document_type']}_{datetime.now().strftime('%Y%m%d%H%M%S')

                    # Determine document category
                    document_category = 'IDENTIFICATION' if row['document_type'] in ['PASSPORT', 'VISA'] else \
                                        'EMPLOYMENT' if row['document_type'] in ['CONTRACT', 'PAYSLIP', 'SALARY', 'POSITION'] e
                                        'IMAGE' if row['document_type'] == 'IMAGE' else \
                                        'OTHER'

                    # Insert into new database
                    conn.execute(text("""
                    INSERT INTO document_metadata (
                        document_id, employee_id, first_name, last_name,
                        email, department_name, document_type, document_category,
                        document_number, original_file_path, upload_date,
                        processed_date, status, is_active
                    ) VALUES (
                        :doc_id, :emp_id, :fname, :lname,
                        :email, :dept, :doc_type, :doc_cat,
                        :doc_num, :orig_path, :upload_date,
                        :proc_date, :status, :active
                    )
                    """), {
                        'doc_id': document_id,
                        'emp_id': row['employee_id'],
                        'fname': row['first_name'],
                        'lname': row['last_name'],
                        'email': row['email'],
                        'dept': row['department_name'],
                        'doc_type': row['document_type'],
                        'doc_cat': document_category,
                        'doc_num': row['document_number'],
                        'orig_path': row['file_path'],
                        'upload_date': row['upload_date'],
                        'proc_date': datetime.now(),
                        'status': 'PENDING_UPLOAD',
                        'active': 1
                    })

                    print(f"Migrated document: {document_id}")
```

```
                # Commit the transaction
                trans.commit()
                print("\nAll records committed successfully!")

            except Exception as e:
                # Rollback in case of error
                trans.rollback()
                print(f"Error during migration, rolling back: {str(e)}")
                raise

        # Verify migration with a new connection
        print("\nVerifying migration...")
        verify_query = """
        SELECT
            document_id,
            first_name,
            last_name,
            document_type,
            status,
            department_name,
            document_category
        FROM document_metadata
        """

        with new_db.connect() as conn:
            result = pd.read_sql(verify_query, conn)
            print("\nMigration Summary:")
            print(f"Total records migrated: {len(result)}")

            if not result.empty:
                print("\nSample of migrated data:")
                print(result.head())

                print("\nDocument categories distribution:")
                print(result['document_category'].value_counts())

                print("\nDocument types distribution:")
                print(result['document_type'].value_counts())

                print("\nStatus distribution:")
                print(result['status'].value_counts())
            else:
                print("No records found in the new database!")

        return True

    except Exception as e:
        print(f"Error during data migration: {str(e)}")
        print("Full error details:", e)
        return False

# Migrate data to new database
migrate_data_to_new_db()
```

```
Starting data migration from old to new database...
Found 9 records to migrate
Migrated document: DOC_1_EXCEL_20250403083552
Migrated document: DOC_1_PDF_20250403083552
Migrated document: DOC_1_IMAGE_20250403083552
Migrated document: DOC_2_WORD_20250403083552
Migrated document: DOC_2_PAYSLIP_20250403083552
Migrated document: DOC_2_IMAGE_20250403083552
Migrated document: DOC_3_POSITION_20250403083552
Migrated document: DOC_3_SALARY_20250403083552
Migrated document: DOC_3_IMAGE_20250403083552

All records committed successfully!

Verifying migration...

Migration Summary:
Total records migrated: 9

Sample of migrated data:
                   document_id first_name last_name document_type  \
0    DOC_1_EXCEL_20250403083552       John       Doe         EXCEL
1      DOC_1_PDF_20250403083552       John       Doe           PDF
2    DOC_1_IMAGE_20250403083552       John       Doe         IMAGE
3     DOC_2_WORD_20250403083552       Jane     Smith          WORD
4  DOC_2_PAYSLIP_20250403083552       Jane     Smith       PAYSLIP

            status department_name document_category
0  PENDING_UPLOAD              HR             OTHER
1  PENDING_UPLOAD              HR             OTHER
2  PENDING_UPLOAD              HR             IMAGE
```

```
3  PENDING_UPLOAD         Finance             OTHER
4  PENDING_UPLOAD         Finance        EMPLOYMENT

Document categories distribution:
document_category
OTHER         3
IMAGE         3
EMPLOYMENT    3
Name: count, dtype: int64

Document types distribution:
document_type
IMAGE       3
EXCEL       1
PDF         1
WORD        1
PAYSLIP     1
POSITION    1
SALARY      1
Name: count, dtype: int64

Status distribution:
status
PENDING_UPLOAD    9
Name: count, dtype: int64
True
```

```
def upload_to_s3():
    try:
        print("Starting S3 upload process...")

        # Initialize S3 client
        s3_client = boto3.client(
            's3',
            aws_access_key_id=AWS_ACCESS_KEY,
            aws_secret_access_key=AWS_SECRET_KEY,
            region_name=AWS_REGION
        )

        # Test S3 connection
        try:
            s3_client.head_bucket(Bucket=BUCKET_NAME)
            print("Successfully connected to S3 bucket!")
        except Exception as e:
            print(f"Error connecting to S3: {str(e)}")
            return False

        # Connect to new database
        new_db = create_engine('sqlite:///new_db.db')

        # Get pending uploads
        with new_db.connect() as conn:
            df = pd.read_sql("SELECT * FROM document_metadata WHERE status = 'PENDING_UPLOAD'", conn)

        print(f"Found {len(df)} documents to upload")

        success_count = 0
        error_count = 0

        for idx, row in df.iterrows():
            try:
                # Check if file exists
                if not os.path.exists(row['original_file_path']):
                    print(f"File not found: {row['original_file_path']}")
                    error_count += 1
                    continue

                # Get file extension
                _, file_extension = os.path.splitext(row['original_file_path'])

                # Create S3 key
                s3_key = f"{row['department_name'].lower()}/{row['document_type'].lower()}/{row['document_id']}{file_extensi

                # Upload to S3
                print(f"Uploading: {row['original_file_path']} to s3://{BUCKET_NAME}/{s3_key}")
                s3_client.upload_file(
                    row['original_file_path'],
                    BUCKET_NAME,
                    s3_key,
                    ExtraArgs={
                        'Metadata': {
                            'employee_id': str(row['employee_id']),
                            'document_type': row['document_type'],
                            'department': row['department_name'],
```

```
                            'document_number': row['document_number']
                        }
                    }
                )

                # Update database
                with new_db.connect() as conn:
                    conn.execute(text("""
                    UPDATE document_metadata
                    SET s3_file_path = :s3_path,
                        status = 'UPLOADED',
                        processed_date = :proc_date
                    WHERE document_id = :doc_id
                    """), {
                        's3_path': f"s3://{BUCKET_NAME}/{s3_key}",
                        'proc_date': datetime.now(),
                        'doc_id': row['document_id']
                    })

                success_count += 1
                print(f"Successfully uploaded: {row['document_id']}")

            except Exception as e:
                error_count += 1
                print(f"Error uploading document {row['document_id']}: {str(e)}")
                continue

        print("\nUpload Summary:")
        print(f"Total documents processed: {len(df)}")
        print(f"Successfully uploaded: {success_count}")
        print(f"Failed: {error_count}")

        # Final status check
        with new_db.connect() as conn:
            status_df = pd.read_sql("""
                SELECT status, COUNT(*) as count
                FROM document_metadata
                GROUP BY status
            """, conn)
            print("\nFinal Status Distribution:")
            print(status_df)

        return True

    except Exception as e:
        print(f"Error during S3 upload: {str(e)}")
        return False

# Upload documents to S3
upload_to_s3()
```

```
Starting S3 upload process...
Successfully connected to S3 bucket!
Found 9 documents to upload
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/Book.xlsx to s3://testempdoc/hr/excel/DOC_1_EXCEL_20250403083
Successfully uploaded: DOC_1_EXCEL_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/details.pdf to s3://testempdoc/hr/pdf/DOC_1_PDF_2025040308355
Successfully uploaded: DOC_1_PDF_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/first_source_jpg.png to s3://testempdoc/hr/image/DOC_1_IMAGE_
Successfully uploaded: DOC_1_IMAGE_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/Dummy.docx to s3://testempdoc/finance/word/DOC_2_WORD_2025040
Successfully uploaded: DOC_2_WORD_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/payslip.pdf to s3://testempdoc/finance/payslip/DOC_2_PAYSLIP_
Successfully uploaded: DOC_2_PAYSLIP_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/firstsource_logo_jpeg.jpeg to s3://testempdoc/finance/image/D
Successfully uploaded: DOC_2_IMAGE_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/position.pdf to s3://testempdoc/operations/position/DOC_3_POS
Successfully uploaded: DOC_3_POSITION_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/salary.pdf to s3://testempdoc/operations/salary/DOC_3_SALARY_
Successfully uploaded: DOC_3_SALARY_20250403083552
Uploading: /Users/espinshalo/Downloads/FDMS/test_documents/firstsource_logo.jpg to s3://testempdoc/operations/image/DOC_
Successfully uploaded: DOC_3_IMAGE_20250403083552

Upload Summary:
Total documents processed: 9
Successfully uploaded: 9
Failed: 0

Final Status Distribution:
           status  count
0  PENDING_UPLOAD      9
True
```

```
def verify_complete_process():
    try:
        print("Verifying complete migration process...")

        # Connect to databases
        old_db = create_engine('sqlite:///old_db.db')
        new_db = create_engine('sqlite:///new_db.db')

        # Check old database
        with old_db.connect() as conn:
            old_count = pd.read_sql("SELECT COUNT(*) as count FROM employee_documents", conn).iloc[0]['count']
            print(f"\nOld database document count: {old_count}")

            print("\nOld database document types:")
            old_types = pd.read_sql("""
                SELECT document_type, COUNT(*) as count
                FROM employee_documents
                GROUP BY document_type
            """, conn)
            print(old_types)

        # Check new database
        with new_db.connect() as conn:
            new_df = pd.read_sql("""
                SELECT
                    document_type,
                    document_category,
                    status,
                    COUNT(*) as count
                FROM document_metadata
                GROUP BY document_type, document_category, status
            """, conn)
            print("\nNew database status summary:")
            print(new_df)

        # Check S3
        s3_client = boto3.client(
            's3',
            aws_access_key_id=AWS_ACCESS_KEY,
            aws_secret_access_key=AWS_SECRET_KEY,
            region_name=AWS_REGION
        )

        try:
            response = s3_client.list_objects_v2(Bucket=BUCKET_NAME)
            s3_count = response.get('KeyCount', 0)
            print(f"\nS3 document count: {s3_count}")

            if 'Contents' in response:
                print("\nS3 files by department:")
                s3_files = {}
                for obj in response['Contents']:
                    dept = obj['Key'].split('/')[0]
                    s3_files[dept] = s3_files.get(dept, 0) + 1

                for dept, count in s3_files.items():
                    print(f"{dept}: {count} files")

                print("\nSample S3 files:")
                for obj in response['Contents'][:5]:
                    print(f"- {obj['Key']}")
        except Exception as e:
            print(f"Error checking S3: {str(e)}")

        return True

    except Exception as e:
        print(f"Error during verification: {str(e)}")
        return False

# Verify complete process
verify_complete_process()
```

```
Verifying complete migration process...

Old database document count: 9

Old database document types:
  document_type  count
0         EXCEL      1
1         IMAGE      3
2       PAYSLIP      1
3           PDF      1
```

```
4      POSITION      1
5        SALARY      1
6          WORD      1

New database status summary:
  document_type document_category          status  count
0         EXCEL             OTHER  PENDING_UPLOAD      1
1         IMAGE             IMAGE  PENDING_UPLOAD      3
2       PAYSLIP        EMPLOYMENT  PENDING_UPLOAD      1
3           PDF             OTHER  PENDING_UPLOAD      1
4      POSITION        EMPLOYMENT  PENDING_UPLOAD      1
5        SALARY        EMPLOYMENT  PENDING_UPLOAD      1
6          WORD             OTHER  PENDING_UPLOAD      1

S3 document count: 29

S3 files by department:
engineering: 3 files
finance: 9 files
hr: 9 files
marketing: 1 files
operations: 7 files

Sample S3 files:
- engineering/payslip/DOC_2_PAYSLIP_20250401112013.txt
- engineering/resignation/DOC_4_RESIGNATION_20250401112013.txt
- engineering/visa/DOC_2_VISA_20250401112013.txt
- finance/contract/DOC_3_CONTRACT_20250401112013.txt
- finance/image/DOC_2_IMAGE_20250403083552.jpeg
True
```

```
def list_s3_files():
    """
    Lists all files in the S3 bucket with complete S3 URLs for downloading.
    """
    try:
        # Create S3 client with your credentials
        s3_client = boto3.client(
            's3',
            aws_access_key_id=AWS_ACCESS_KEY,
            aws_secret_access_key=AWS_SECRET_KEY,
            region_name=AWS_REGION
        )

        # S3 bucket details
        bucket_name = BUCKET_NAME
        region = AWS_REGION

        print("Listing all files in S3 bucket with download URLs...")
        print("-" * 80)

        # List all objects in the bucket
        paginator = s3_client.get_paginator('list_objects_v2')
        pages = paginator.paginate(Bucket=bucket_name)

        # Dictionary to store files by department
        files_by_dept = {}

        for page in pages:
            if 'Contents' in page:
                for obj in page['Contents']:
                    # Get the key (file path)
                    key = obj['Key']

                    # Skip if it's a directory marker
                    if key.endswith('/'):
                        continue

                    # Extract department and document type from the path
                    parts = key.split('/')
                    if len(parts) >= 2:
                        dept = parts[0]
                        doc_type = parts[1] if len(parts) > 1 else 'Unknown'

                        # Generate the complete S3 URL
                        s3_url = f"https://{bucket_name}.s3.{region}.amazonaws.com/{key}"

                        if dept not in files_by_dept:
                            files_by_dept[dept] = {}
                        if doc_type not in files_by_dept[dept]:
                            files_by_dept[dept][doc_type] = []

                        files_by_dept[dept][doc_type].append({
                            'file_name': parts[-1],
                            'size': obj['Size'],
```

```
                            'size': obj['Size'],
                            'last_modified': obj['LastModified'],
                            's3_path': key,
                            'download_url': s3_url
                        })

        # Print the organized list
        for dept in sorted(files_by_dept.keys()):
            print(f"\nDepartment: {dept.upper()}")
            print("=" * 50)

            for doc_type in sorted(files_by_dept[dept].keys()):
                print(f"\nDocument Type: {doc_type}")
                print("-" * 30)

                for file_info in files_by_dept[dept][doc_type]:
                    print(f"File: {file_info['file_name']}")
                    print(f"Size: {file_info['size']} bytes")
                    print(f"Last Modified: {file_info['last_modified']}")
                    print(f"S3 Path: {file_info['s3_path']}")
                    print(f"Download URL: {file_info['download_url']}")
                    print()

        # Print summary
        total_files = sum(len(files)
                          for dept in files_by_dept.values()
                          for files in dept.values())

        print("\nSummary:")
        print(f"Total Departments: {len(files_by_dept)}")
        print(f"Total Files: {total_files}")

        # Save URLs to a CSV file for easy reference
        url_data = []
        for dept in files_by_dept:
            for doc_type in files_by_dept[dept]:
                for file_info in files_by_dept[dept][doc_type]:
                    url_data.append({
                        'Department': dept,
                        'Document Type': doc_type,
                        'File Name': file_info['file_name'],
                        'S3 Path': file_info['s3_path'],
                        'Download URL': file_info['download_url'],
                        'Last Modified': file_info['last_modified']
                    })

        df = pd.DataFrame(url_data)
        csv_filename = 's3_file_urls.csv'
        df.to_csv(csv_filename, index=False)
        print(f"\nURLs have been saved to {csv_filename}")

    except Exception as e:
        print(f"Error listing S3 files: {str(e)}")

# Run the function
list_s3_files()
```

```
S3 Path: operations/position/DOC_3_POSITION_20250403083552.pdf
Download URL: https://testempdoc.s3.ap-south-1.amazonaws.com/operations/position/DOC_3_POSITION_20250403083552.pdf


Document Type: salary
--------------------------------
File: DOC_3_SALARY_20250401114446.pdf
Size: 102 bytes
Last Modified: 2025-04-01 11:45:08+00:00
S3 Path: operations/salary/DOC_3_SALARY_20250401114446.pdf
Download URL: https://testempdoc.s3.ap-south-1.amazonaws.com/operations/salary/DOC_3_SALARY_20250401114446.pdf

File: DOC_3_SALARY_20250403072945.pdf
Size: 102 bytes
Last Modified: 2025-04-03 07:30:05+00:00
S3 Path: operations/salary/DOC_3_SALARY_20250403072945.pdf
Download URL: https://testempdoc.s3.ap-south-1.amazonaws.com/operations/salary/DOC_3_SALARY_20250403072945.pdf

File: DOC_3_SALARY_20250403083552.pdf
Size: 102 bytes
Last Modified: 2025-04-03 08:37:13+00:00
S3 Path: operations/salary/DOC_3_SALARY_20250403083552.pdf
Download URL: https://testempdoc.s3.ap-south-1.amazonaws.com/operations/salary/DOC_3_SALARY_20250403083552.pdf


Summary:
Total Departments: 5
Total Files: 29

URLs have been saved to s3_file_urls.csv
```

Start coding or generate with AI.