Here is the complete, self-contained documentation for **Project 1**. You can save this as a Markdown file (e.g., Project_1_Migration.md) or print it as a PDF to work offline.

---

# PROJECT 1: THE LEGACY MIGRATION

**Role:** Data Engineer

**Difficulty:** Easy (Foundation)

**Est. Time:** 2-4 Hours

---

## 1. Executive Summary & Scenario

**The Business Problem:**

"QuickMart," a regional retail chain, currently relies on a legacy Point-of-Sale (POS) system. Every night at midnight, this system exports the day's sales into a CSV file on a local server.

Currently, the Analytics team manually copies this file to their laptops to run Excel reports. This is error-prone, unsecure, and slow.

**The Goal:**

Automate this process. You must build a pipeline that:

1. Picks up the CSV file from the "On-Premise" server (simulated by your local machine).
2. Uploads it to the Cloud (Azure Data Lake Storage).
3. Loads it into a structured SQL Database for reporting.

**The Constraint (Senior Level Requirement):**

Your solution must be **Environment Aware**. You are building this in a **DEV** environment today, but it must be designed so that we can deploy it to **PROD** tomorrow without rewriting the code.

---

## 2. Architecture Overview

We are implementing a classic **EL (Extract-Load)** pattern. We will not be doing complex transformations (ETL) yet; our priority is data accessibility.

**The Stack:**

- **Source:** Local Python Script (Simulating the POS System).
- **Storage (Staging):** Azure Data Lake Gen2 (ADLS) - Container: raw-bronze.
- **Orchestration:** Azure Data Factory (ADF).
- **Destination:** Azure SQL Database.

---

## 3. The Data Generator (Source)

Since we don't have access to a real POS system, we will simulate one.

**Action:** Copy the code below into a file named generate_sales.py on your computer. Run it to generate your source data.

Python

```python
import csv
import random
import datetime

# Configuration
NUM_ROWS = 1000
FILENAME = "sales_data.csv"

def generate_data():
    print(f"Generating {FILENAME} with {NUM_ROWS} rows...")
    with open(FILENAME, mode='w', newline='') as file:
        writer = csv.writer(file)
        # Header
        writer.writerow(['TransactionID', 'ProductID', 'CustomerID', 'TransactionDate', 'Quantity',
'Amount'])

        # Data
        for i in range(1, NUM_ROWS + 1):
            transaction_id = i
            product_id = random.randint(100, 150)
            customer_id = random.randint(1, 500)

            # Random date within last 30 days
            start_date = datetime.date.today() - datetime.timedelta(days=30)
        random_date = start_date + datetime.timedelta(days=random.randint(0, 30))

        quantity = random.randint(1, 10)
        # Random amount between $10 and $500
        amount = round(random.uniform(10.0, 500.0), 2)

        writer.writerow([transaction_id, product_id, customer_id, random_date, quantity,
amount])

    print(f"✅ Success! Generated {FILENAME}.")

if __name__ == "__main__":
    generate_data()
```

---

## 4. The Design Phase (Interactive)

Before you touch the Azure Portal, you must act as the Architect. Read the challenges below and write down your answers. Then, check the "Solution" section to validate your thinking.

### Challenge A: The Data Model (DDL)

**Task:** Write the SQL CREATE TABLE statement for the destination table dbo.Sales_Raw.

- **Critical Thinking:** Look at the Amount field in the CSV. It represents money. What data type should you use? (Hint: FLOAT is often a bad choice for finance).
- **Critical Thinking:** How do you handle TransactionDate?

### Challenge B: The Environment Strategy (CI/CD)

**Task:** You have sql-quickmart-dev and sql-quickmart-prod. How do you configure ADF so it doesn't break when we move to Prod?

- **Scenario:** If you hardcode the connection string Server=tcp:dev-sql... inside ADF, the Prod pipeline will still point to Dev.
- **Question:** What feature of ADF allows us to make this dynamic?

### Challenge C: The Test Plan

**Task:** After the pipeline runs, how do you prove it worked?

- Write 2 SQL queries you will run to verify the data is accurate.

---

### 🛑 STOP! Attempt the design above before scrolling down.

## ✅ Architect's Solution (Reference)

**Answer A: The Table Schema**

SQL

```sql
CREATE TABLE dbo.Sales_Raw (
    TransactionID INT PRIMARY KEY,
    ProductID INT NOT NULL,
    CustomerID INT,
    TransactionDate DATE,      -- Or DATETIME if time is included
    Quantity INT,
    Amount DECIMAL(10, 2)      -- CORRECT: DECIMAL or MONEY.
                               -- INCORRECT: FLOAT (Causes rounding errors).
);
```

**Answer B: Environment Strategy**

We use **Parameters** and **Key Vault**.

1. We create an **Azure Key Vault** to store the connection strings as "Secrets" (DbConnectionString-Dev, DbConnectionString-Prod).
2. In ADF, we create a **Global Parameter** called Environment (set to "Dev").
3. In the Linked Service, we reference the Key Vault Secret dynamically: @{concat('DbConnectionString-', pipeline().globalParameters.Environment)}
   *Note: For this beginner project, if Key Vault is too complex, just use ADF Parameters for the Server Name.*

**Answer C: The Test Queries**

1. **Completeness:** SELECT COUNT(*) FROM dbo.Sales_Raw; (Must match CSV row count).
2. **Accuracy:** SELECT SUM(Amount) FROM dbo.Sales_Raw; (Must match Excel sum).

---

## 5. Implementation Guide (Step-by-Step)

Now that the design is solid, let's build it.

### Phase 1: Azure Setup

1. **Resource Group:** Create rg-quickmart-project1.
2. **Storage:** Create an **Azure Data Lake Storage Gen2** account (adlsquickmartdev).

    - *Important:* In the "Advanced" tab, enable **Hierarchical Namespace**.
    - Create a container named raw-bronze.
    - **Upload** your sales_data.csv into this container manually.
3. **Database:** Create an **Azure SQL Database** (sql-quickmart-dev).
    - Use the **Basic** or **Serverless** tier to keep costs near zero.
    - Open the "Query Editor" in the portal and run your CREATE TABLE script (from Answer A).

## Phase 2: Azure Data Factory (ADF) Configuration

1. **Create ADF:** Create a Data Factory resource (adf-quickmart-dev).
2. **Open Studio:** Click "Launch Studio".

### Step 2.1: Linked Services (Connections)

Go to the **Manage** tab (Briefcase icon) > **Linked Services** > **New**.

1. **Azure Data Lake Gen2:**
    - Name: LS_ADLS_Main.
    - Auth: Account Key (simplest) or System Assigned Managed Identity (best practice).
2. **Azure SQL Database:**
    - Name: LS_SQL_Main.
    - Enter your server name and SQL authentication credentials.
    - *Pro Tip:* This is where you would use Key Vault in a real project.

### Step 2.2: Datasets (Data Shape)

Go to the **Author** tab (Pencil icon) > **Datasets** > **New**.

1. **Source Dataset:**
    - Select **Azure Data Lake Gen2** -> **DelimitedText (CSV)**.
    - Name: DS_Raw_Sales_CSV.
    - Linked Service: LS_ADLS_Main.
    - File Path: raw-bronze/sales_data.csv.
    - **Check:** "First row as header".
2. **Sink Dataset:**
    - Select **Azure SQL Database**.
    - Name: DS_SQL_Sales_Table.
    - Linked Service: LS_SQL_Main.
    - Table: dbo.Sales_Raw.

### Step 2.3: The Pipeline

1. Create a new Pipeline named PL_Ingest_Sales_Data.
2. Drag the **Copy Data** activity onto the canvas.
3. **Source Tab:** Select DS_Raw_Sales_CSV.
4. **Sink Tab:** Select DS_SQL_Sales_Table.

5. **Mapping Tab:** Click **Import Schemas**.
   - Verify that TransactionDate maps to TransactionDate.
   - Verify that Amount (String/Double) maps to Amount (Decimal).
6. **Settings Tab:** (Optional) Enable "Fault Tolerance" to skip incompatible rows if you want to test error handling.

---

## 6. Execution & Verification

1. **Debug:** Click the **Debug** button at the top of the pipeline canvas.
2. **Monitor:** Watch the "Output" tab. Wait for the green checkmark.
3. **Verify:**
   - Go back to your Azure SQL Database Query Editor.
   - Run your Test Queries (from Answer C).
   - *Success:* You should see exactly 1,000 rows (or however many your script generated).

---

## 7. Key Learnings & Takeaways

- **Copy Activity vs. Data Flow:** We used Copy Activity because we were just *moving* data. If we needed to calculate Profit = Amount - Cost during the move, we would have used a **Mapping Data Flow** (which uses Spark).
- **Linked Services:** These are the "Connection Strings" of ADF. Decoupling them from specific environments (Dev/Prod) is the first step in CI/CD.
- **Data Types:** Using DECIMAL for money is a non-negotiable rule in Data Engineering.

**Status:** Once you have successfully verified the row counts, you have completed Project 1.

**Next Up:** Project 2 - The API Collector (Handling JSON and Serverless SQL).

This is the final step of Project 1. In a real-world scenario, before you deploy to production, you would go through a **Design Review** or **Thesis Defense** with a Lead Architect.

I will act as the Lead Architect. I am going to ask you **5 Critical Questions** about the solution you just built. These aren't just "quiz questions"; they are scenarios that determine if your pipeline will survive in production.

**Instructions:** Answer these questions in your own words. Be honest if you don't know—I will explain the "Senior Level" answer after you try.

---

## Project 1 Defense: The Inquisition

### Q1. The "Double Run" Problem (Idempotency)

*"I just accidentally clicked the 'Trigger Now' button twice. Your pipeline ran two times in a row with the same `sales_data.csv` file. **What does the data in the SQL table look like now?** Do we have duplicates? If so, how would you fix the pipeline to prevent this in the future?"*

### Q2. The Cost Justification

*"Why did you choose **Azure SQL Database** for this? My friend told me **Azure Synapse Analytics** is the 'big data' tool. Why didn't we use Synapse Dedicated Pool? Explain your choice based on cost and data volume."*

### Q3. The "Bad Data" Scenario

*"Tomorrow, the source system glitches. The CSV file arrives, but row #500 has a text string `"ERROR"` in the `Amount` column instead of a number. **What happens when your pipeline runs?** Does it fail completely? Does it load 499 rows? How would you configure ADF to log this error but load the rest?"*

### Q4. Security & Access

*"Currently, your ADF connects to the Database using a username and password saved in the Linked Service. **If I fire you today and disable your account, does the pipeline stop working?** If not, why is this a security risk, and what is the Azure feature we should use to abstract these credentials?"*

### Q5. The Architecture Pivot

*"We used a **Copy Activity**. If I asked you to also calculate `TotalProfit = (Price - Cost) * Quantity` while moving the data, **could you still use Copy Activity?** If not, which ADF activity would you switch to, and what is the trade-off (downside) of making that switch?"*

---

**Reply with your answers (e.g., "A1: ...").** Once you submit your defense, I will grade you and provide the "Architect's detailed answer key" to close out Project 1.