# LLM Comparison for COBOL code conversion

## Comparison Table

|  |  |  |
| --- | --- | --- |
|  | OpenAI **GPT-4** | Google **Gemini** |
| Model Card | Model Name: **gpt-4-0613**  Context Window: 8,192 tokens  Model Name: **gpt-4-32k-0613**  Context Window: 32,768 tokens  Model Name: **gpt-4-1106-preview (NEW)**  Context Window: 128,000 tokens | Model Name: **Gemini-Pro**  Context Window: 32,768 tokens |
| Pricing | Input: $0.03 / 1K tokens  Output: $0.06 / 1K tokens | Free to use right now, within limits.  Limit: 60 QPM (Queries per minute)  Input: Free  Output: Free |
| COBOL Batch to Python  [Appendix - 1.1](#_COBOL_Batch_program)  Complexity - Simple | Generated Python code  [Appendix - 4](#_GPT-4_Convert_Simple)  Pros:   * Able to capture all program flow and business logic * Used functional programming style * Able to understand fixed width copybook format and parse using offset and length   Cons:   * Ignore field data type declaration, assumed all fields as String * COBOL program does not have any error handling, Python code followed suite without error handling | Generated Python code  [Appendix - 5](#_Gemini-Pro_Convert_Simple)  Pros:   * Able to capture all program flow and business logic * Able to understand fixed width copybook format and parse using offset, length and data type   Cons:   * Generated code is procedural * Python code does not have any error handling |
| COBOL DB2 Batch to Python  [Appendix - 1.2](#_COBOL_Batch_DB2)  Complexity - Simple | Generated Python code  [Appendix - 6](#_GPT-4_Convert_Simple_1)  Pros:   * Able to capture all program flow and business logic * Used ORM (package – sqlalchemy) to access to database * Used functional programming style   Cons:   * No exception handling | Generated Python  [Appendix - 7](#_Gemini-Pro_Convert_Simple_1)  Pros:   * Able to capture all program flow and business logic * Used ORM (package – sqlalchemy) to access to database   Cons:   * Generated code is procedural * No exception handling |
| CICS COBOL Map to Python MVC Application (Django framework)  [Appendix – 1.3](#_1.3_CICS_Map)  Complexity - Simple | Generated Python code  [Appendix - 9](#_GPT-4_Convert_CICS)  Pros:   * Able to convert BMS map into HTML template * Scaffolded the project structure   Cons:   * Missed some CALL and CICS LINK statements * Created more of stub python controller code * Requires more than 60% manual work to make it runnable | Generated Python  [Appendix - 8](#_Gemini-Pro_Convert_CICS)  Pros:   * Able to convert BMS map into HTML template * Scaffolded the project structure   Cons:   * Missed some CALL and CICS LINK statements * Created more of stub python controller code   Requires more than 60% manual work to make it runnable |

## 

## Appendix

### COBOL Source Codes

#### COBOL Batch program – Record processing (Complexity – Simple)

       IDENTIFICATION DIVISION.

       PROGRAM-ID.  PINVRPT.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       ENVIRONMENT DIVISION.

       INPUT-OUTPUT SECTION.

       FILE-CONTROL.

           SELECT INALL ASSIGN TO PINV.

           SELECT INACT ASSIGN TO PACT.

           SELECT INSPN ASSIGN TO PSPN.

           SELECT INDES ASSIGN TO PDES.

           SELECT RPTALL ASSIGN TO RPTALL.

           SELECT RPTACT ASSIGN TO RPTACT.

           SELECT RPTSPN ASSIGN TO RPTSPN.

           SELECT RPTDES ASSIGN TO RPTDES.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       DATA DIVISION.

       FILE SECTION.

       FD INALL

           RECORDING MODE F.

       01  INALL-RECORD-ALL.

           05  GRAIN-SERIAL-ALL             PIC 9(12).

           05  GRAIN-TYPE-ALL               PIC X(10).

           05  GRAIN-FORMULA-ALL            PIC X(15).

           05  GRAIN-STATUS-ALL             PIC X(1).

           05  GRAIN-QA-ALL                 PIC X(4).

           05  GRAIN-QUALITY-ALL            PIC 9(2).

           05  FILLER                       PIC 9(10).

           05  FILLER                       PIC 9(10).

           05  GRAIN-WEIGHT-ALL             PIC 9(10).

           05  FILLER                       PIC X(375).

       FD INACT

           RECORDING MODE F.

       01  INALL-RECORD-ACT.

           05  GRAIN-SERIAL-ACT             PIC 9(12).

           05  GRAIN-TYPE-ACT               PIC X(10).

           05  GRAIN-FORMULA-ACT            PIC X(15).

           05  GRAIN-STATUS-ACT             PIC X(1).

           05  GRAIN-QA-ACT                 PIC X(4).

           05  GRAIN-QUALITY-ACT            PIC 9(2).

           05  FILLER                       PIC 9(10).

           05  FILLER                       PIC 9(10).

           05  GRAIN-WEIGHT-ACT             PIC 9(10).

           05  FILLER                       PIC X(375).

       FD INSPN

           RECORDING MODE F.

       01  INALL-RECORD-SPN.

           05  GRAIN-SERIAL-SPN             PIC 9(12).

           05  GRAIN-TYPE-SPN               PIC X(10).

           05  GRAIN-FORMULA-SPN            PIC X(15).

           05  GRAIN-STATUS-SPN             PIC X(1).

           05  GRAIN-QA-SPN                 PIC X(4).

           05  GRAIN-QUALITY-SPN            PIC 9(2).

           05  FILLER                       PIC 9(10).

           05  FILLER                       PIC 9(10).

           05  GRAIN-WEIGHT-SPN             PIC 9(10).

           05  FILLER                       PIC X(375).

       FD INDES

           RECORDING MODE F.

       01  INALL-RECORD-DES.

           05  GRAIN-SERIAL-DES             PIC 9(12).

           05  GRAIN-TYPE-DES               PIC X(10).

           05  GRAIN-FORMULA-DES            PIC X(15).

           05  GRAIN-STATUS-DES             PIC X(1).

           05  GRAIN-QA-DES                 PIC X(4).

           05  GRAIN-QUALITY-DES            PIC 9(2).

           05  FILLER                       PIC 9(10).

           05  FILLER                       PIC 9(10).

           05  GRAIN-WEIGHT-DES             PIC 9(10).

           05  FILLER                       PIC X(375).

       FD RPTALL

           RECORDING MODE V.

       01  OUTFILE-ALL                 PIC X(200).

       FD RPTACT

           RECORDING MODE V.

       01  OUTFILE-ACTIVE              PIC X(200).

       FD RPTSPN

           RECORDING MODE V.

       01  OUTFILE-SPENT               PIC X(200).

       FD RPTDES

           RECORDING MODE V.

       01  OUTFILE-DESTROYED           PIC X(200).

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       WORKING-STORAGE SECTION.

       01  WS-CURRENT-DATE.

           05  WS-YEAR                 PIC 9(4).

           05  WS-MONTH                PIC 9(2).

           05  WS-DAY                  PIC 9(2).

           05  WS-HOURS                PIC 9(2).

           05  WS-MINUTES              PIC 9(2).

           05  WS-SECONDS              PIC 9(2).

           05  WS-HUND-SECOND          PIC 9(2).

           05  WS-GMT                  PIC X(5).

       01  OUT-RECORD.

           05  SERIAL-OUT              PIC X(12).

           05  FILLER                  PIC X(2)

               VALUE  SPACES.

           05  STATUS-OUT              PIC X(10).

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  TYPE-OUT                PIC X(10).

           05  FILLER                  PIC X(2)

               VALUE  SPACES.

           05  FORMULA-OUT             PIC X(15).

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  QA-OUT                  PIC X(4).

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  WEIGHT-OUT              PIC X(10).

           05  FILLER                  PIC X

               VALUE 'G'.

       01  DATE-LINE.

           05  FILLER                  PIC X(12)

               VALUE 'REPORT DATE:'.

           05  FILLER                  PIC X(1)

               VALUE SPACES.

           05  DT-YEAR                 PIC 9(4).

           05  FILLER                  PIC X

               VALUE '/'.

           05  DT-MONTH                PIC 9(2).

           05  FILLER                  PIC X

               VALUE '/'.

           05  DT-DAY                  PIC 9(2).

       01  HEADER-1.

           05  FILLER                  PIC X(100)

               VALUE  'HAYNIE RESEARCH & DEVELOPMENT'.

       01  HEADER-ALL.

           05  FILLER                  PIC X(100)

               VALUE  'PROPELLANT GRAIN INVENTORY REPORT - ALL'.

       01  HEADER-ACTIVE.

           05  FILLER                  PIC X(100)

               VALUE  'PROPELLANT GRAIN INVENTORY REPORT - ACTIVE'.

       01  HEADER-SPENT.

           05  FILLER                  PIC X(100)

               VALUE  'PROPELLANT GRAIN INVENTORY REPORT - SPENT'.

       01  HEADER-DESTROYED.

           05  FILLER                  PIC X(100)

               VALUE  'PROPELLANT GRAIN INVENTORY REPORT - DESTROYED'.

       01  HEADER-3.

           05  FILLER                  PIC X(14)

               VALUE  'SERIAL'.

           05  FILLER                  PIC X(12)

               VALUE  'STATUS'.

           05  FILLER                  PIC X(12)

               VALUE  'TYPE'.

           05  FILLER                  PIC X(17)

               VALUE  'FORMULA'.

           05  FILLER                  PIC X(6)

               VALUE  'QA'.

           05  FILLER                  PIC X(11)

               VALUE  'WEIGHT'.

       01  HEADER-4.

           05  FILLER                  PIC X(12)

               VALUE  '------------'.

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  FILLER                  PIC X(10)

               VALUE  '----------'.

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  FILLER                  PIC X(10)

               VALUE '----------'.

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  FILLER                  PIC X(15)

               VALUE '---------------'.

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  FILLER                  PIC X(4)

               VALUE '----'.

           05  FILLER                  PIC X(2)

               VALUE SPACES.

           05  FILLER                  PIC X(11)

               VALUE '-----------'.

       01  SWITCHES.

           05  INALL-EOF-SWITCH         PIC X(1) VALUE 'N'.

           05  INACT-EOF-SWITCH         PIC X(1) VALUE 'N'.

           05  INSPN-EOF-SWITCH         PIC X(1) VALUE 'N'.

           05  INDES-EOF-SWITCH         PIC X(1) VALUE 'N'.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       PROCEDURE DIVISION.

       MAIN-PROGRAM.

           MOVE FUNCTION CURRENT-DATE TO WS-CURRENT-DATE.

           MOVE WS-YEAR TO DT-YEAR.

           MOVE WS-MONTH TO DT-MONTH.

           MOVE WS-DAY TO DT-DAY.

           PERFORM REPORT-ALL.

           PERFORM REPORT-ACTIVE.

           PERFORM REPORT-SPENT.

           PERFORM REPORT-DESTROYED.

           STOP RUN.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       REPORT-ALL.

           OPEN INPUT INALL

                OUTPUT RPTALL.

           READ INALL

               AT END

                   MOVE 'Y' TO INALL-EOF-SWITCH

           END-READ.

           MOVE HEADER-1 TO OUTFILE-ALL.

           WRITE OUTFILE-ALL.

           MOVE HEADER-ALL TO OUTFILE-ALL.

           WRITE OUTFILE-ALL.

           MOVE DATE-LINE TO OUTFILE-ALL.

           WRITE OUTFILE-ALL.

           MOVE SPACES TO OUTFILE-ALL.

           WRITE OUTFILE-ALL.

           MOVE HEADER-3 TO OUTFILE-ALL.

           WRITE OUTFILE-ALL.

           MOVE HEADER-4 TO OUTFILE-ALL.

           WRITE OUTFILE-ALL.

           PERFORM ALL-LOOP

               UNTIL INALL-EOF-SWITCH = 'Y'.

           CLOSE INALL

                 RPTALL.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       REPORT-ACTIVE.

           OPEN INPUT INACT

                OUTPUT RPTACT.

           READ INACT

               AT END

                   MOVE 'Y' TO INACT-EOF-SWITCH

           END-READ.

           MOVE HEADER-1 TO OUTFILE-ACTIVE.

           WRITE OUTFILE-ACTIVE.

           MOVE HEADER-ACTIVE TO OUTFILE-ACTIVE.

           WRITE OUTFILE-ACTIVE.

           MOVE DATE-LINE TO OUTFILE-ACTIVE.

           WRITE OUTFILE-ACTIVE.

           MOVE SPACES TO OUTFILE-ACTIVE.

           WRITE OUTFILE-ACTIVE.

           MOVE HEADER-3 TO OUTFILE-ACTIVE.

           WRITE OUTFILE-ACTIVE.

           MOVE HEADER-4 TO OUTFILE-ACTIVE.

           WRITE OUTFILE-ACTIVE.

           PERFORM ACTIVE-LOOP

               UNTIL INACT-EOF-SWITCH = 'Y'.

           CLOSE INACT

                 RPTACT.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       REPORT-SPENT.

           OPEN INPUT INSPN

                OUTPUT RPTSPN.

           READ INSPN

               AT END

                   MOVE 'Y' TO INSPN-EOF-SWITCH

           END-READ.

           MOVE HEADER-1 TO OUTFILE-SPENT.

           WRITE OUTFILE-SPENT.

           MOVE HEADER-SPENT TO OUTFILE-SPENT.

           WRITE OUTFILE-SPENT.

           MOVE DATE-LINE TO OUTFILE-SPENT.

           WRITE OUTFILE-SPENT.

           MOVE SPACES TO OUTFILE-SPENT.

           WRITE OUTFILE-SPENT.

           MOVE HEADER-3 TO OUTFILE-SPENT.

           WRITE OUTFILE-SPENT.

           MOVE HEADER-4 TO OUTFILE-SPENT.

           WRITE OUTFILE-SPENT.

           PERFORM SPENT-LOOP

               UNTIL INSPN-EOF-SWITCH = 'Y'.

           CLOSE INSPN

                 RPTSPN.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       REPORT-DESTROYED.

           OPEN INPUT INDES

                OUTPUT RPTDES.

           READ INDES

               AT END

                   MOVE 'Y' TO INDES-EOF-SWITCH

           END-READ.

           MOVE HEADER-1 TO OUTFILE-DESTROYED.

           WRITE OUTFILE-DESTROYED.

           MOVE HEADER-DESTROYED TO OUTFILE-DESTROYED.

           WRITE OUTFILE-DESTROYED.

           MOVE DATE-LINE TO OUTFILE-DESTROYED.

           WRITE OUTFILE-DESTROYED.

           MOVE SPACES TO OUTFILE-DESTROYED.

           WRITE OUTFILE-DESTROYED.

           MOVE HEADER-3 TO OUTFILE-DESTROYED.

           WRITE OUTFILE-DESTROYED.

           MOVE HEADER-4 TO OUTFILE-DESTROYED.

           WRITE OUTFILE-DESTROYED.

           PERFORM DESTROYED-LOOP

               UNTIL INDES-EOF-SWITCH = 'Y'.

           CLOSE INDES

                 RPTDES.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       ALL-LOOP.

           MOVE GRAIN-SERIAL-ALL TO SERIAL-OUT.

           IF GRAIN-STATUS-ALL = 0

               MOVE "ACTIVE" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-ALL = 1

               MOVE "SPENT" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-ALL = 3

               MOVE "DESTROYED" TO STATUS-OUT

           ELSE

               MOVE "OTHER" TO STATUS-OUT

           END-IF.

           MOVE GRAIN-TYPE-ALL TO TYPE-OUT.

           MOVE GRAIN-FORMULA-ALL TO FORMULA-OUT.

           MOVE GRAIN-QA-ALL TO QA-OUT.

           MOVE GRAIN-WEIGHT-ALL TO WEIGHT-OUT.

           MOVE OUT-RECORD TO OUTFILE-ALL.

           WRITE OUTFILE-ALL.

           READ INALL

               AT END

                   MOVE 'Y' TO INALL-EOF-SWITCH

           END-READ.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       ACTIVE-LOOP.

           MOVE GRAIN-SERIAL-ACT TO SERIAL-OUT.

           IF GRAIN-STATUS-ACT = 0

               MOVE "ACTIVE" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-ACT = 1

               MOVE "SPENT" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-ACT = 3

               MOVE "DESTROYED" TO STATUS-OUT

           ELSE

               MOVE "OTHER" TO STATUS-OUT

           END-IF.

           MOVE GRAIN-TYPE-ACT TO TYPE-OUT.

           MOVE GRAIN-FORMULA-ACT TO FORMULA-OUT.

           MOVE GRAIN-QA-ACT TO QA-OUT.

           MOVE GRAIN-WEIGHT-ACT TO WEIGHT-OUT.

           MOVE OUT-RECORD TO OUTFILE-ACTIVE.

           WRITE OUTFILE-ACTIVE.

           READ INACT

               AT END

                   MOVE 'Y' TO INACT-EOF-SWITCH

           END-READ.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       SPENT-LOOP.

           MOVE GRAIN-SERIAL-SPN TO SERIAL-OUT.

           IF GRAIN-STATUS-SPN = 0

               MOVE "ACTIVE" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-SPN = 1

               MOVE "SPENT" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-SPN = 3

               MOVE "DESTROYED" TO STATUS-OUT

           ELSE

               MOVE "OTHER" TO STATUS-OUT

           END-IF.

           MOVE GRAIN-TYPE-SPN TO TYPE-OUT.

           MOVE GRAIN-FORMULA-SPN TO FORMULA-OUT.

           MOVE GRAIN-QA-SPN TO QA-OUT.

           MOVE GRAIN-WEIGHT-SPN TO WEIGHT-OUT.

           MOVE OUT-RECORD TO OUTFILE-SPENT.

           WRITE OUTFILE-SPENT.

           READ INSPN

               AT END

                   MOVE 'Y' TO INSPN-EOF-SWITCH

           END-READ.

      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

       DESTROYED-LOOP.

           MOVE GRAIN-SERIAL-DES TO SERIAL-OUT.

           IF GRAIN-STATUS-DES = 0

               MOVE "ACTIVE" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-DES = 1

               MOVE "SPENT" TO STATUS-OUT

           ELSE IF GRAIN-STATUS-DES = 3

               MOVE "DESTROYED" TO STATUS-OUT

           ELSE

               MOVE "OTHER" TO STATUS-OUT

           END-IF.

           MOVE GRAIN-TYPE-DES TO TYPE-OUT.

           MOVE GRAIN-FORMULA-DES TO FORMULA-OUT.

           MOVE GRAIN-QA-DES TO QA-OUT.

           MOVE GRAIN-WEIGHT-DES TO WEIGHT-OUT.

           MOVE OUT-RECORD TO OUTFILE-DESTROYED.

           WRITE OUTFILE-DESTROYED.

           READ INDES

               AT END

                   MOVE 'Y' TO INDES-EOF-SWITCH

           END-READ.

#### COBOL Batch DB2 program (Complexity – Simple)

       IDENTIFICATION DIVISION.

       PROGRAM-ID. LDB2SAMP.

       DATA DIVISION.

       WORKING-STORAGE SECTION.

       EXEC SQL INCLUDE SQLCA END-EXEC.

      \*COPY EMPLOYEE.

       01 EMPLOYEE.

         05 EMPNO PIC X(6).

         05 FIRSTNME.

           49 LEN PIC S9(4) COMP-5.

           49 DAT PIC X(12).

         05 MIDINIT PIC X(1).

         05 LASTNAME.

           49 LEN PIC S9(4) COMP-5.

           49 DAT PIC X(15).

         05 WORKDEPT PIC X(3).

         05 PHONENO PIC X(4).

         05 HIREDATE PIC X(10).

         05 JOB PIC X(8).

         05 EDLEVEL PIC S9(4) COMP-5.

         05 SEX PIC X(1).

         05 BIRTHDATE PIC X(10).

         05 SALARY PIC S9(7)V9(2) COMP-3.

         05 BONUS PIC S9(7)V9(2) COMP-3.

         05 COMM PIC S9(7)V9(2) COMP-3.

       01 VARIABLES.

           02 RECORDCOUNT        PIC S9(8) USAGE COMP-3.

           02 DISPLAYRC          PIC S9(4) USAGE DISPLAY.

           02 OUT                PIC X(200).

           02 NEXTEMP            PIC X(6).

           02 NEXTEMPNO          PIC 9(6).

           02 FIRSTNMEP          PIC X(12).

           02 LASTNAMEP          PIC X(15).

       LINKAGE SECTION.

       PROCEDURE DIVISION.

           MOVE LOW-VALUES TO VARIABLES.

           EXEC SQL SELECT COUNT (\*)

                           INTO :RECORDCOUNT

                           FROM EMPLOYEE

                           END-EXEC.

           IF SQLCODE = 0 THEN

               MOVE RECORDCOUNT TO DISPLAYRC

               MOVE "THE NUMBER OF EMPLOYEES IN THE DATABASE IS " TO OUT

               MOVE DISPLAYRC TO OUT(44:4)

           ELSE IF SQLCODE = 100 THEN

               MOVE "NO EMPLOYEES FOUND IN DATABASE" TO OUT

           ELSE

               MOVE "SQL ERROR " TO OUT

               MOVE SQLCODE TO OUT(11:10)

           END-IF.

           DISPLAY OUT.

           IF SQLCODE = 0 THEN

               EXEC SQL SELECT MAX (EMPNO)

                               INTO :NEXTEMP

                               FROM EMPLOYEE

                               END-EXEC

               MOVE NEXTEMP TO NEXTEMPNO

               COMPUTE NEXTEMPNO = NEXTEMPNO + 10

               MOVE NEXTEMPNO TO EMPNO

               MOVE "FRANK" TO FIRSTNMEP

               MOVE "JONES" TO LASTNAMEP

               MOVE "Y" TO MIDINIT

               MOVE "A00" TO WORKDEPT

               MOVE "1234" TO PHONENO

               MOVE "04-30-1979" TO HIREDATE

               MOVE "Clerk" TO JOB

               MOVE 15 TO EDLEVEL

               MOVE "M" TO SEX

               MOVE "05-30-1954" TO BIRTHDATE

               MOVE "36170" TO SALARY

               MOVE "400" TO BONUS

               MOVE "2387" TO COMM

               EXEC SQL INSERT INTO EMPLOYEE

                               VALUES(:EMPNO,

                                      :FIRSTNMEP,

                                      :MIDINIT,

                                      :LASTNAMEP,

                                      :WORKDEPT,

                                      :PHONENO,

                                      :HIREDATE,

                                      :JOB,

                                      :EDLEVEL,

                                      :SEX,

                                      :BIRTHDATE,

                                      :SALARY,

                                      :BONUS,

                                      :COMM)

                               END-EXEC

               IF SQLCODE = 0 THEN

                  EXEC SQL SELECT \*

                           INTO :EMPNO,

                                :FIRSTNME,

                                :MIDINIT,

                                :LASTNAME,

                                :WORKDEPT,

                                :PHONENO,

                                :HIREDATE,

                                :JOB,

                                :EDLEVEL,

                                :SEX,

                                :BIRTHDATE,

                                :SALARY,

                                :BONUS,

                                :COMM

                           FROM EMPLOYEE

                           WHERE EMPNO = :EMPNO

                           END-EXEC

                  EXEC SQL COMMIT END-EXEC

                  MOVE FIRSTNMEP TO OUT

                  MOVE LASTNAMEP TO OUT(14:15)

                  DISPLAY OUT

                  DISPLAY "ADDED TO THE DATABASE"

               ELSE

                  MOVE "SQL ERROR" TO OUT

                  DISPLAY OUT

               END-IF

           END-IF.

           GOBACK.

#### 1.3 CICS Map COBOL program and MAP

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 00010000

EPSMORT    DFHMSD TYPE=&SYSPARM,MODE=INOUT,LANG=COBOL,              @P3CX0008000

               STORAGE=AUTO,TIOAPFX=YES,DSATTS=(COLOR,HILIGHT),        X00090000

               MAPATTS=(COLOR,HILIGHT)                                  00091000

\*       MENU MAP..

EPMENU  DFHMDI SIZE=(24,80),CTRL=(PRINT,FREEKB)                         00110000

LITLOAN  DFHMDF POS=(8,15),LENGTH=23,INITIAL='Amount of Loan:',        \*

               ATTRB=(ASKIP,NORM)

EPLOAN   DFHMDF POS=(8,42),LENGTH=12,INITIAL='999999999.99',           \*

               ATTRB=(NUM,FSET,IC,NORM)

         DFHMDF POS=(8,55),                                            \*

               ATTRB=ASKIP

LITYEARS DFHMDF POS=(9,15),LENGTH=24,                                  \*

               INITIAL='Length of Loan in Years:',                     \*

               ATTRB=(PROT,NORM)

EPYEARS  DFHMDF POS=(9,42),LENGTH=2,INITIAL='99',                      \*

               ATTRB=(NUM,FSET,NORM),COLOR=GREEN

         DFHMDF POS=(9,45),LENGTH=0,                                   \*

               ATTRB=ASKIP

EPDIFF1  DFHMDF POS=(10,15),LENGTH=22,INITIAL='Interest Rate: ',       \*

               ATTRB=(PROT,NORM)

EPRATE   DFHMDF POS=(10,42),LENGTH=5,INITIAL='99.99',                  \*

               ATTRB=(NUM,FSET,NORM),COLOR=GREEN

         DFHMDF POS=(15,7),LENGTH=60,                                  \*

               INITIAL='Press PF9 to see companies that can match or be\*

               at this rate',                                          \*

               ATTRB=(ASKIP,NORM),HILIGHT=OFF,COLOR=BLUE

         DFHMDF POS=(14,15),LENGTH=43,                                 \*

               INITIAL='Press F3 to quit or Enter to calculate loan',  \*

               ATTRB=(ASKIP,NORM),HILIGHT=OFF,COLOR=BLUE

EPDIFF2  DFHMDF POS=(17,15),LENGTH=22,INITIAL='Monthly Payment: ',     \*

               ATTRB=(PROT,NORM)

EPPAYMNT DFHMDF POS=(17,42),LENGTH=12,                                 \*

               ATTRB=(PROT,NORM),HILIGHT=OFF,COLOR=YELLOW

MSGERR  DFHMDF POS=(24,15),ATTRB=(PROT,DRK),LENGTH=40,                 X00340000

               INITIAL='INVALID KEY PRESSED'                            00350000

         DFHMDF POS=(2,48),LENGTH=0,                                   \*

               ATTRB=ASKIP

         DFHMDF POS=(5,25),LENGTH=26,INITIAL='EPS MORTGAGE CALCULATOR',\*

               ATTRB=(ASKIP,BRT)

EPSMORT  DFHMSD TYPE=FINAL                                               0036000

        END                                                             00370000

       ID DIVISION.

       PROGRAM-ID. EPSCMORT.

      \*    THIS DEMONSTRATES CICS/DEBUG           - EPSDEMOS 2008

      \*

      \*    THIS PROGRAM WILL RECEIVE A DATE AND COVERT THE DATE TO

      \*    AN INTEGER IN A CALLED PROGRAM TO DETERMINE DAYS FROM

      \*    CURRENT DATE.

      \*

      \*    (C) 2017 IBM - JIM HILDNER RESERVED.

       ENVIRONMENT DIVISION.

       CONFIGURATION SECTION.

       SOURCE-COMPUTER. IBM-FLEX-ES.

       OBJECT-COMPUTER. IBM-FLEX-ES.

      \*

       DATA DIVISION.

       WORKING-STORAGE SECTION.

      \*

       01  W-FLAGS.

           10  W-SEND-FLAG                    PIC X.

               88  SEND-ERASE                   VALUE '1'.

               88  SEND-DATAONLY                VALUE '2'.

               88  SEND-MAPONLY                 VALUE '3'.

               88  SEND-DATAONLY-ALARM          VALUE '4'.

               88  SEND-ALL                     VALUE '5'.

       01 W-CONVERSIONS.

           05  W-PMT-CNVRT     PIC X(12).

           05  W-PMT-NUMBER

               REDEFINES W-PMT-CNVRT

                               PIC 9(10)V99.

           05  WS-FORMAT-NUMBER PIC Z,ZZZ,ZZ9.99.

           05  W-PRINC-CNVRT   PIC X(12).

           05  W-PRINC-NUMBER

               REDEFINES W-PRINC-CNVRT

                               PIC 9(10)V99.

       01 W-CALL-PROGRAM                      PIC X(8).

      \*

       01 W-RETIREMENT-WA                     PIC 9(4).

       01 W-COMAREA-LENGTH                    PIC 9(4) COMP.

       01  SQL-ERROR-MSG.

           03  FILLER              PIC X(11)      VALUE 'SQL ERROR: '.

           03  SQL-ERROR-CODE      PIC 9(5) DISPLAY.

      \*

           EXEC SQL

               INCLUDE SQLCA

           END-EXEC.

      \*

           EXEC SQL DECLARE SYSIBM.SYSDUMMY1 TABLE

           ( IBMREQD                        CHAR(1) NOT NULL

           ) END-EXEC.

      \*

       01 IBMREQD                           PIC X(1).

      \*

       01  END-OF-TRANS-MSG                 PIC X(30)

             VALUE 'END OF TRANSACTION - THANK YOU'.

       01  BLANK-MSG                        PIC X(1) VALUE ' '.

           COPY DFHAID.

      \*    COPY DFHEIBLK.

           COPY EPSMORT.

       01  W-COMMUNICATION-AREA.

           COPY EPSMTCOM.

       COPY EPSNBRPM.

       LINKAGE SECTION.

       01 DFHCOMMAREA.

       COPY EPSMTCOM.

       PROCEDURE DIVISION USING DFHCOMMAREA.

       EPSCMORT-MAINLINE.

      \* Call procedure to do SQL call

           PERFORM A805-DUMMY-SQL-CALL

           MOVE LENGTH OF DFHCOMMAREA to W-COMAREA-LENGTH.

           MOVE DFHCOMMAREA to W-COMMUNICATION-AREA.

           EVALUATE TRUE

               WHEN EIBCALEN = ZERO

      \* First time in - Show Screen

                   MOVE LOW-VALUES TO EPMENUO

                   SET SEND-ERASE TO TRUE

                   PERFORM A300-SEND-MAP

                   MOVE '3' TO

                      PROCESS-INDICATOR OF W-COMMUNICATION-AREA

               WHEN EIBAID = DFHCLEAR

      \* Process CLEAR key

                   MOVE LOW-VALUES TO EPMENUO

                   SET SEND-ERASE TO TRUE

                   PERFORM A300-SEND-MAP

               WHEN EIBAID = DFHPF3 OR DFHPF12

      \* Process END/RETURN keys

                  IF PROCESS-INDICATOR OF W-COMMUNICATION-AREA = '3'

                      EXEC CICS

                         SEND TEXT FROM (END-OF-TRANS-MSG)

**ERASE**

                         FREEKB

                      END-EXEC

                      EXEC CICS

                           RETURN

                      END-EXEC

                   ELSE

                      SET SEND-ALL TO TRUE

                      EXEC CICS

                         SEND TEXT FROM (BLANK-MSG)

**ERASE**

                         FREEKB

                      END-EXEC

                      PERFORM A300-SEND-MAP

                      MOVE '3' TO

                          PROCESS-INDICATOR OF W-COMMUNICATION-AREA

                   END-IF

               WHEN EIBAID = DFHPF9

                   MOVE '9' TO

                      PROCESS-INDICATOR OF W-COMMUNICATION-AREA

                   EXEC CICS LINK PROGRAM( 'EPSMLIST' )

                          COMMAREA( W-COMMUNICATION-AREA )

                   END-EXEC

               WHEN EIBAID = DFHENTER

      \* Process ENTER Key

                   IF PROCESS-INDICATOR OF W-COMMUNICATION-AREA = '3'

                      PERFORM A100-PROCESS-MAP

                   ELSE

                      EXEC CICS LINK PROGRAM('EPSMLIST')

                             COMMAREA( W-COMMUNICATION-AREA )

                      END-EXEC

                   END-IF

               WHEN OTHER

      \* Process Data

                    IF PROCESS-INDICATOR OF W-COMMUNICATION-AREA = '3'

                      PERFORM A600-CALCULATE-MORTGAGE

                      EXEC CICS RETURN END-EXEC

      \*             ELSE

      \*                MOVE X'E8' TO MSGERRA

      \*                MOVE LOW-VALUES TO EPMENUO

      \*                SET SEND-DATAONLY-ALARM TO TRUE

      \*                PERFORM A300-SEND-MAP

                    END-IF

           END-EVALUATE

           EXEC CICS

               RETURN TRANSID('EPSP')

               COMMAREA(W-COMMUNICATION-AREA)

               LENGTH(W-COMAREA-LENGTH)

           END-EXEC.

       A100-PROCESS-MAP.

           PERFORM A400-RECEIVE-MAP.

           PERFORM A600-CALCULATE-MORTGAGE

           SET SEND-DATAONLY TO TRUE

           PERFORM A300-SEND-MAP

               .

       A300-SEND-MAP.

           EVALUATE TRUE

              WHEN SEND-MAPONLY

                   EXEC CICS

                     SEND MAP ('EPMENU')

                       MAPSET('EPSMORT')

                       MAPONLY

                       CURSOR

                   END-EXEC

              WHEN SEND-ERASE

                   EXEC CICS

                     SEND MAP ('EPMENU')

                         MAPSET('EPSMORT')

                         FROM(EPMENUO)

**ERASE**

                         CURSOR

                   END-EXEC

              WHEN SEND-DATAONLY

                   EXEC CICS

                     SEND MAP ('EPMENU')

                         MAPSET('EPSMORT')

                         FROM(EPMENUO)

                         DATAONLY

                         CURSOR

                   END-EXEC

              WHEN SEND-ALL

                   EXEC CICS

                     SEND MAP ('EPMENU')

                         MAPSET('EPSMORT')

                         FROM(EPMENUO)

                     END-EXEC.

       A400-RECEIVE-MAP.

           EXEC CICS

                RECEIVE MAP('EPMENU')

                   MAPSET('EPSMORT')

                   INTO (EPMENUI)

           END-EXEC.

           MOVE EPLOANI        TO EPSPARM-VALIDATE-DATA.

           MOVE LENGTH OF EPLOANI

                               TO EPSPARM-MAX-LENGTH.

           CALL 'EPSNBRVL' USING EPS-NUMBER-VALIDATION.

           COMPUTE EPSPCOM-PRINCIPLE-DATA

                OF W-COMMUNICATION-AREA

                = EPSPARM-NUMBER + EPSPARM-DECIMAL.

           MOVE EPYEARSI             TO EPSPARM-VALIDATE-DATA.

           MOVE LENGTH OF EPYEARSI   TO EPSPARM-MAX-LENGTH.

           CALL 'EPSNBRVL' USING EPS-NUMBER-VALIDATION.

           COMPUTE EPSPCOM-NUMBER-OF-YEARS

                OF W-COMMUNICATION-AREA

                = EPSPARM-NUMBER + EPSPARM-DECIMAL.

           MOVE EPRATEI              TO EPSPARM-VALIDATE-DATA.

           MOVE LENGTH OF EPRATEI    TO EPSPARM-MAX-LENGTH.

           CALL 'EPSNBRVL' USING EPS-NUMBER-VALIDATION.

           COMPUTE EPSPCOM-QUOTED-INTEREST-RATE

                OF W-COMMUNICATION-AREA

                = EPSPARM-NUMBER + EPSPARM-DECIMAL.

       A600-CALCULATE-MORTGAGE.

           MOVE 'Y' TO EPSPCOM-YEAR-MONTH-IND

                           OF W-COMMUNICATION-AREA.

           MOVE 'EPSCSMRT' TO W-CALL-PROGRAM

           EXEC CICS LINK PROGRAM( W-CALL-PROGRAM )

                          COMMAREA( W-COMMUNICATION-AREA )

           END-EXEC

           .

           MOVE EPSPCOM-RETURN-MONTH-PAYMENT

                             OF W-COMMUNICATION-AREA

                             TO WS-FORMAT-NUMBER.

           MOVE WS-FORMAT-NUMBER

                             TO EPPAYMNTO.

           MOVE EPSPCOM-ERRMSG

                             OF W-COMMUNICATION-AREA

                             TO MSGERRO.

       A805-DUMMY-SQL-CALL.

           EXEC SQL

               SELECT IBMREQD

                    INTO :IBMREQD

                    FROM SYSIBM.SYSDUMMY1

           END-EXEC.

      \*

           IF SQLCODE = 100

               MOVE 'No rows found on SYSDUMM1.' TO MSGERRO

           ELSE

               IF SQLCODE NOT = 0

                   MOVE SQLCODE TO SQL-ERROR-CODE

                   MOVE SQL-ERROR-MSG TO MSGERRO

               END-IF

           END-IF.

      \*

### OpenAI GPT-4 Python SDK Usage

from openai import OpenAI

from IPython.display import Markdown

import os

client = OpenAI(

    api\_key=OPENAI\_KEY,

    organization=ORG\_ID

)

messages = [

    {

        "role": "system",

        "content": instruction

    },

    {

        "role": "user",

        "content": f'"""{cobol\_source}"""',

    }

]

chat\_completion = client.chat.completions.create(

    messages=messages,

    model="gpt-4",

)

Markdown(chat\_completion.choices[0].message.content)

### Google Gemini-Pro Python SDK Usage

import google.generativeai as genai

from IPython.display import Markdown

import os

genai.configure(api\_key = GOOGLE\_API\_KEY)

model = genai.GenerativeModel('gemini-pro')

response = model.generate\_content(

    prompt,

    generation\_config=genai.types.GenerationConfig(

        candidate\_count=1,

        max\_output\_tokens=4000,

    )

)

Markdown(response.text)

### GPT-4 Convert Simple COBOL Batch Program (Appendix 1.1) to Python

*Temperature:* 0.2

*Prompt:*

You are a helpful AI assistant.

You convert Cobol code into Python code. Please do not explain the code, just provide the code.

Provide a main method to run the application. Also do not omit any code for brevity. We want to see the whole code.

Hints:

\* Input and Output files are fixed width flat files.

\* Input file records does not have any delimited characters, access the fields by offset and length as per File descriptor definition.

User input Cobol code is delimited by triple quotes. Convert the entire Cobol code to Python.

*Token Usage:*

CompletionUsage(completion\_tokens=758, prompt\_tokens=3870, total\_tokens=4628)

*Output Python Code:*

import datetime

def main():

    current\_date = datetime.datetime.now()

    report\_all()

    report\_active()

    report\_spent()

    report\_destroyed()

def report\_all():

    with open("PINV", "r") as inall, open("RPTALL", "w") as rptall:

        for line in inall:

            grain\_serial\_all = line[0:12]

            grain\_status\_all = line[37:38]

            grain\_type\_all = line[12:22]

            grain\_formula\_all = line[22:37]

            grain\_qa\_all = line[38:42]

            grain\_weight\_all = line[62:72]

            status\_out = get\_status(grain\_status\_all)

            write\_to\_file(

                rptall,

                grain\_serial\_all,

                status\_out,

                grain\_type\_all,

                grain\_formula\_all,

                grain\_qa\_all,

                grain\_weight\_all,

            )

def report\_active():

    with open("PACT", "r") as inact, open("RPTACT", "w") as rptact:

        for line in inact:

            grain\_serial\_act = line[0:12]

            grain\_status\_act = line[37:38]

            grain\_type\_act = line[12:22]

            grain\_formula\_act = line[22:37]

            grain\_qa\_act = line[38:42]

            grain\_weight\_act = line[62:72]

            status\_out = get\_status(grain\_status\_act)

            write\_to\_file(

                rptact,

                grain\_serial\_act,

                status\_out,

                grain\_type\_act,

                grain\_formula\_act,

                grain\_qa\_act,

                grain\_weight\_act,

            )

def report\_spent():

    with open("PSPN", "r") as inspn, open("RPTSPN", "w") as rptspn:

        for line in inspn:

            grain\_serial\_spn = line[0:12]

            grain\_status\_spn = line[37:38]

            grain\_type\_spn = line[12:22]

            grain\_formula\_spn = line[22:37]

            grain\_qa\_spn = line[38:42]

            grain\_weight\_spn = line[62:72]

            status\_out = get\_status(grain\_status\_spn)

            write\_to\_file(

                rptspn,

                grain\_serial\_spn,

                status\_out,

                grain\_type\_spn,

                grain\_formula\_spn,

                grain\_qa\_spn,

                grain\_weight\_spn,

            )

def report\_destroyed():

    with open("PDES", "r") as indes, open("RPTDES", "w") as rptdes:

        for line in indes:

            grain\_serial\_des = line[0:12]

            grain\_status\_des = line[37:38]

            grain\_type\_des = line[12:22]

            grain\_formula\_des = line[22:37]

            grain\_qa\_des = line[38:42]

            grain\_weight\_des = line[62:72]

            status\_out = get\_status(grain\_status\_des)

            write\_to\_file(

                rptdes,

                grain\_serial\_des,

                status\_out,

                grain\_type\_des,

                grain\_formula\_des,

                grain\_qa\_des,

                grain\_weight\_des,

            )

def get\_status(status):

    if status == "0":

        return "ACTIVE"

    elif status == "1":

        return "SPENT"

    elif status == "3":

        return "DESTROYED"

    else:

        return "OTHER"

def write\_to\_file(file, serial, status, type, formula, qa, weight):

    file.write(f"{serial}  {status}  {type}  {formula}  {qa}  {weight}G\n")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

### Gemini-Pro Convert Simple COBOL Batch Program (Appendix 1.1) to Python

*Prompt:*

You are a helpful AI assistant.

You convert Cobol code into Python code. Please do not explain the code, just provide the code.

Provide a main method to run the application. Also do not omit any code for brevity. We want to see the whole code.

Hints:

\* Input and Output files are fixed width flat files.

\* Input file records does not have any delimited characters, access the fields by offset and length as per File descriptor definition.

COBOL code to be converted to Python is given below delimited by ###.

###

{cobol\_source}

###

*Token Usage:*

Input tokens=4889

*Output Python Code*:

import datetime

# Define the input and output files

input\_file\_all = "PINV"

input\_file\_act = "PACT"

input\_file\_spn = "PSPN"

input\_file\_des = "PDES"

output\_file\_all = "RPTALL"

output\_file\_act = "RPTACT"

output\_file\_spn = "RPTSPN"

output\_file\_des = "RPTDES"

# Define the record layouts

input\_record\_all = [

    ("GRAIN-SERIAL-ALL", 12, "9"),

    ("GRAIN-TYPE-ALL", 10, "X"),

    ("GRAIN-FORMULA-ALL", 15, "X"),

    ("GRAIN-STATUS-ALL", 1, "X"),

    ("GRAIN-QA-ALL", 4, "X"),

    ("GRAIN-QUALITY-ALL", 2, "9"),

    ("FILLER", 10, "9"),

    ("FILLER", 10, "9"),

    ("GRAIN-WEIGHT-ALL", 10, "9"),

    ("FILLER", 375, "X"),

]

input\_record\_act = [

    ("GRAIN-SERIAL-ACT", 12, "9"),

    ("GRAIN-TYPE-ACT", 10, "X"),

    ("GRAIN-FORMULA-ACT", 15, "X"),

    ("GRAIN-STATUS-ACT", 1, "X"),

    ("GRAIN-QA-ACT", 4, "X"),

    ("GRAIN-QUALITY-ACT", 2, "9"),

    ("FILLER", 10, "9"),

    ("FILLER", 10, "9"),

    ("GRAIN-WEIGHT-ACT", 10, "9"),

    ("FILLER", 375, "X"),

]

input\_record\_spn = [

    ("GRAIN-SERIAL-SPN", 12, "9"),

    ("GRAIN-TYPE-SPN", 10, "X"),

    ("GRAIN-FORMULA-SPN", 15, "X"),

    ("GRAIN-STATUS-SPN", 1, "X"),

    ("GRAIN-QA-SPN", 4, "X"),

    ("GRAIN-QUALITY-SPN", 2, "9"),

    ("FILLER", 10, "9"),

    ("FILLER", 10, "9"),

    ("GRAIN-WEIGHT-SPN", 10, "9"),

    ("FILLER", 375, "X"),

]

input\_record\_des = [

    ("GRAIN-SERIAL-DES", 12, "9"),

    ("GRAIN-TYPE-DES", 10, "X"),

    ("GRAIN-FORMULA-DES", 15, "X"),

    ("GRAIN-STATUS-DES", 1, "X"),

    ("GRAIN-QA-DES", 4, "X"),

    ("GRAIN-QUALITY-DES", 2, "9"),

    ("FILLER", 10, "9"),

    ("FILLER", 10, "9"),

    ("GRAIN-WEIGHT-DES", 10, "9"),

    ("FILLER", 375, "X"),

]

output\_record = [

    ("SERIAL-OUT", 12, "X"),

    ("FILLER", 2, "X", " "),

    ("STATUS-OUT", 10, "X"),

    ("FILLER", 2, "X", " "),

    ("TYPE-OUT", 10, "X"),

    ("FILLER", 2, "X", " "),

    ("FORMULA-OUT", 15, "X"),

    ("FILLER", 2, "X", " "),

    ("QA-OUT", 4, "X"),

    ("FILLER", 2, "X", " "),

    ("WEIGHT-OUT", 10, "X"),

    ("FILLER", 1, "X", "G"),

]

# Define the header and footer records

date\_line = "REPORT DATE: {}/{}/{}".format(

    datetime.datetime.now().year,

    datetime.datetime.now().month,

    datetime.datetime.now().day,

)

header\_1 = "HAYNIE RESEARCH & DEVELOPMENT"

header\_all = "PROPELLANT GRAIN INVENTORY REPORT - ALL"

header\_active = "PROPELLANT GRAIN INVENTORY REPORT - ACTIVE"

header\_spent = "PROPELLANT GRAIN INVENTORY REPORT - SPENT"

header\_destroyed = "PROPELLANT GRAIN INVENTORY REPORT - DESTROYED"

header\_3 = (

    "SERIAL".ljust(14)

    + "STATUS".ljust(12)

    + "TYPE".ljust(12)

    + "FORMULA".ljust(17)

    + "QA".ljust(6)

    + "WEIGHT".ljust(11)

)

header\_4 = (

    "------------".ljust(12)

    + "----------".ljust(10)

    + "----------".ljust(10)

    + "---------------".ljust(15)

    + "----".ljust(4)

    + "-----------".ljust(11)

)

# Define the switches to control the loops

inall\_eof\_switch = "N"

inact\_eof\_switch = "N"

inspn\_eof\_switch = "N"

indes\_eof\_switch = "N"

# Open the input and output files

with open(input\_file\_all, "r") as inall, open(input\_file\_act, "r") as inact, open(

    input\_file\_spn, "r"

) as inspn, open(input\_file\_des, "r") as indes, open(

    output\_file\_all, "w"

) as outall, open(

    output\_file\_act, "w"

) as outact, open(

    output\_file\_spn, "w"

) as outspn, open(

    output\_file\_des, "w"

) as outdes:

    # Write the header records

    outall.write(header\_1 + "\n")

    outall.write(header\_all + "\n")

    outall.write(date\_line + "\n")

    outall.write("\n")

    outall.write(header\_3 + "\n")

    outall.write(header\_4 + "\n")

    outact.write(header\_1 + "\n")

    outact.write(header\_active + "\n")

    outact.write(date\_line + "\n")

    outact.write("\n")

    outact.write(header\_3 + "\n")

    outact.write(header\_4 + "\n")

    outspn.write(header\_1 + "\n")

    outspn.write(header\_spent + "\n")

    outspn.write(date\_line + "\n")

    outspn.write("\n")

    outspn.write(header\_3 + "\n")

    outspn.write(header\_4 + "\n")

    outdes.write(header\_1 + "\n")

    outdes.write(header\_destroyed + "\n")

    outdes.write(date\_line + "\n")

    outdes.write("\n")

    outdes.write(header\_3 + "\n")

    outdes.write(header\_4 + "\n")

    # Read the first record from each input file

    inall\_record = inall.readline()

    inact\_record = inact.readline()

    inspn\_record = inspn.readline()

    indes\_record = indes.readline()

    # Loop through the input files until all records have been processed

    while inall\_record or inact\_record or inspn\_record or indes\_record:

        # Process the ALL input file

        if inall\_record:

            (

                grain\_serial\_all,

                grain\_type\_all,

                grain\_formula\_all,

                grain\_status\_all,

                grain\_qa\_all,

                grain\_quality\_all,

                \_,

                \_,

                grain\_weight\_all,

                \_,

            ) = input\_record\_all

            status\_out = (

                "ACTIVE"

                if grain\_status\_all == 0

                else "SPENT"

                if grain\_status\_all == 1

                else "DESTROYED"

                if grain\_status\_all == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_all

            output\_record[5][1] = grain\_formula\_all

            output\_record[7][1] = grain\_qa\_all

            output\_record[9][1] = grain\_weight\_all

            outall.write("".join([field[1] for field in output\_record]) + "\n")

            inall\_record = inall.readline()

        if inall\_eof\_switch == "Y":

            break

        # Process the ACTIVE input file

        if inact\_record:

            (

                grain\_serial\_act,

                grain\_type\_act,

                grain\_formula\_act,

                grain\_status\_act,

                grain\_qa\_act,

                grain\_quality\_act,

                \_,

                \_,

                grain\_weight\_act,

                \_,

            ) = input\_record\_act

            status\_out = (

                "ACTIVE"

                if grain\_status\_act == 0

                else "SPENT"

                if grain\_status\_act == 1

                else "DESTROYED"

                if grain\_status\_act == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_act

            output\_record[5][1] = grain\_formula\_act

            output\_record[7][1] = grain\_qa\_act

            output\_record[9][1] = grain\_weight\_act

            outact.write("".join([field[1] for field in output\_record]) + "\n")

            inact\_record = inact.readline()

        if inact\_eof\_switch == "Y":

            break

        # Process the SPENT input file

        if inspn\_record:

            (

                grain\_serial\_spn,

                grain\_type\_spn,

                grain\_formula\_spn,

                grain\_status\_spn,

                grain\_qa\_spn,

                grain\_quality\_spn,

                \_,

                \_,

                grain\_weight\_spn,

                \_,

            ) = input\_record\_spn

            status\_out = (

                "ACTIVE"

                if grain\_status\_spn == 0

                else "SPENT"

                if grain\_status\_spn == 1

                else "DESTROYED"

                if grain\_status\_spn == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_spn

            output\_record[5][1] = grain\_formula\_spn

            output\_record[7][1] = grain\_qa\_spn

            output\_record[9][1] = grain\_weight\_spn

            outspn.write("".join([field[1] for field in output\_record]) + "\n")

            inspn\_record = inspn.readline()

        if inspn\_eof\_switch == "Y":

            break

        # Process the DESTROYED input file

        if indes\_record:

            (

                grain\_serial\_des,

                grain\_type\_des,

                grain\_formula\_des,

                grain\_status\_des,

                grain\_qa\_des,

                grain\_quality\_des,

                \_,

                \_,

                grain\_weight\_des,

                \_,

            ) = input\_record\_des

            status\_out = (

                "ACTIVE"

                if grain\_status\_des == 0

                else "SPENT"

                if grain\_status\_des == 1

                else "DESTROYED"

                if grain\_status\_des == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_des

            output\_record[5][1] = grain\_formula\_des

            output\_record[7][1] = grain\_qa\_des

            output\_record[9][1] = grain\_weight\_des

            outdes.write("".join([field[1] for field in output\_record]) + "\n")

            indes\_record = indes.readline()

        if indes\_eof\_switch == "Y":

            break

    # Close the input and output files

    inall.close()

    inact.close()

    inspn.close()

    indes.close()

    outall.close()

    outact.close()

    outspn.close()

    outdes.close()

# Define the main function to run the program

def main():

    # Call the report generation functions

    report\_all()

    report\_active()

    report\_spent()

    report\_destroyed()

# Define the report generation functions

def report\_all():

    # Open the input and output files

    with open(input\_file\_all, "r") as inall, open(output\_file\_all, "w") as outall:

        # Write the header records

        outall.write(header\_1 + "\n")

        outall.write(header\_all + "\n")

        outall.write(date\_line + "\n")

        outall.write("\n")

        outall.write(header\_3 + "\n")

        outall.write(header\_4 + "\n")

        # Read the first record from the input file

        inall\_record = inall.readline()

        # Loop through the input file until all records have been processed

        while inall\_record:

            # Process the input record

            (

                grain\_serial\_all,

                grain\_type\_all,

                grain\_formula\_all,

                grain\_status\_all,

                grain\_qa\_all,

                grain\_quality\_all,

                \_,

                \_,

                grain\_weight\_all,

                \_,

            ) = input\_record\_all

            status\_out = (

                "ACTIVE"

                if grain\_status\_all == 0

                else "SPENT"

                if grain\_status\_all == 1

                else "DESTROYED"

                if grain\_status\_all == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_all

            output\_record[5][1] = grain\_formula\_all

            output\_record[7][1] = grain\_qa\_all

            output\_record[9][1] = grain\_weight\_all

            outall.write("".join([field[1] for field in output\_record]) + "\n")

            # Read the next record from the input file

            inall\_record = inall.readline()

def report\_active():

    # Open the input and output files

    with open(input\_file\_act, "r") as inact, open(output\_file\_act, "w") as outact:

        # Write the header records

        outact.write(header\_1 + "\n")

        outact.write(header\_active + "\n")

        outact.write(date\_line + "\n")

        outact.write("\n")

        outact.write(header\_3 + "\n")

        outact.write(header\_4 + "\n")

        # Read the first record from the input file

        inact\_record = inact.readline()

        # Loop through the input file until all records have been processed

        while inact\_record:

            # Process the input record

            (

                grain\_serial\_act,

                grain\_type\_act,

                grain\_formula\_act,

                grain\_status\_act,

                grain\_qa\_act,

                grain\_quality\_act,

                \_,

                \_,

                grain\_weight\_act,

                \_,

            ) = input\_record\_act

            status\_out = (

                "ACTIVE"

                if grain\_status\_act == 0

                else "SPENT"

                if grain\_status\_act == 1

                else "DESTROYED"

                if grain\_status\_act == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_act

            output\_record[5][1] = grain\_formula\_act

            output\_record[7][1] = grain\_qa\_act

            output\_record[9][1] = grain\_weight\_act

            outact.write("".join([field[1] for field in output\_record]) + "\n")

            # Read the next record from the input file

            inact\_record = inact.readline()

def report\_spent():

    # Open the input and output files

    with open(input\_file\_spn, "r") as inspn, open(output\_file\_spn, "w") as outspn:

        # Write the header records

        outspn.write(header\_1 + "\n")

        outspn.write(header\_spent + "\n")

        outspn.write(date\_line + "\n")

        outspn.write("\n")

        outspn.write(header\_3 + "\n")

        outspn.write(header\_4 + "\n")

        # Read the first record from the input file

        inspn\_record = inspn.readline()

        # Loop through the input file until all records have been processed

        while inspn\_record:

            # Process the input record

            (

                grain\_serial\_spn,

                grain\_type\_spn,

                grain\_formula\_spn,

                grain\_status\_spn,

                grain\_qa\_spn,

                grain\_quality\_spn,

                \_,

                \_,

                grain\_weight\_spn,

                \_,

            ) = input\_record\_spn

            status\_out = (

                "ACTIVE"

                if grain\_status\_spn == 0

                else "SPENT"

                if grain\_status\_spn == 1

                else "DESTROYED"

                if grain\_status\_spn == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_spn

            output\_record[5][1] = grain\_formula\_spn

            output\_record[7][1] = grain\_qa\_spn

            output\_record[9][1] = grain\_weight\_spn

            outspn.write("".join([field[1] for field in output\_record]) + "\n")

            # Read the next record from the input file

            inspn\_record = inspn.readline()

def report\_destroyed():

    # Open the input and output files

    with open(input\_file\_des, "r") as indes, open(output\_file\_des, "w") as outdes:

        # Write the header records

        outdes.write(header\_1 + "\n")

        outdes.write(header\_destroyed + "\n")

        outdes.write(date\_line + "\n")

        outdes.write("\n")

        outdes.write(header\_3 + "\n")

        outdes.write(header\_4 + "\n")

        # Read the first record from the input file

        indes\_record = indes.readline()

        # Loop through the input file until all records have been processed

        while indes\_record:

            # Process the input record

            (

                grain\_serial\_des,

                grain\_type\_des,

                grain\_formula\_des,

                grain\_status\_des,

                grain\_qa\_des,

                grain\_quality\_des,

                \_,

                \_,

                grain\_weight\_des,

                \_,

            ) = input\_record\_des

            status\_out = (

                "ACTIVE"

                if grain\_status\_des == 0

                else "SPENT"

                if grain\_status\_des == 1

                else "DESTROYED"

                if grain\_status\_des == 3

                else "OTHER"

            )

            output\_record[1][1] = status\_out

            output\_record[3][1] = grain\_type\_des

            output\_record[5][1] = grain\_formula\_des

            output\_record[7][1] = grain\_qa\_des

            output\_record[9][1] = grain\_weight\_des

            outdes.write("".join([field[1] for field in output\_record]) + "\n")

            # Read the next record from the input file

            indes\_record = indes.readline()

# Call the main function to run the program

if \_\_name\_\_ == "\_\_main\_\_":

    main()

### GPT-4 Convert Simple COBOL Batch DB2 Program (Appendix 1.2) to Python

*Temperature:* 0.2

*Prompt:*

You are a helpful AI assistant.

You convert Cobol code into Python code. Please do not explain the code, just provide the code.

Provide a main method to run the application. Also do not omit any code for brevity. We want to see the whole code.

Hints:

\* Cobol code accesses DB2 database tables via EXEC SQL..END-EXEC statements

\* Assume equivalent Oracle table exists with similar schema as the DB2 tables

\* Access Oracle table via SQLAlchemy package in python

User input Cobol code is delimited by triple quotes. Convert the entire Cobol code to Python.

*Token Usage:*

CompletionUsage(completion\_tokens=442, prompt\_tokens=1105, total\_tokens=1547)

*Output Python Code:*

from sqlalchemy import create\_engine, Table, MetaData

from sqlalchemy.sql import select, func, text

from sqlalchemy.exc import SQLAlchemyError

engine = create\_engine("oracle+cx\_oracle://user:password@localhost/dbname")

metadata = MetaData()

employee = Table("EMPLOYEE", metadata, autoload\_with=engine)

def main():

    conn = engine.connect()

    try:

        record\_count = conn.execute(

            select([func.count()]).select\_from(employee)

        ).scalar()

        if record\_count is not None:

            print(f"THE NUMBER OF EMPLOYEES IN THE DATABASE IS {record\_count}")

        else:

            print("NO EMPLOYEES FOUND IN DATABASE")

        max\_empno = conn.execute(select([func.max(employee.c.EMPNO)])).scalar()

        if max\_empno is not None:

            next\_empno = int(max\_empno) + 10

            new\_employee = {

                "EMPNO": str(next\_empno),

                "FIRSTNME": "FRANK",

                "MIDINIT": "Y",

                "LASTNAME": "JONES",

                "WORKDEPT": "A00",

                "PHONENO": "1234",

                "HIREDATE": "04-30-1979",

                "JOB": "Clerk",

                "EDLEVEL": 15,

                "SEX": "M",

                "BIRTHDATE": "05-30-1954",

                "SALARY": "36170",

                "BONUS": "400",

                "COMM": "2387",

            }

            conn.execute(employee.insert().values(new\_employee))

            result = conn.execute(

                select([employee]).where(employee.c.EMPNO == str(next\_empno))

            ).first()

            if result is not None:

                print(

                    f"{result['FIRSTNME']} {result['LASTNAME']} ADDED TO THE DATABASE"

                )

            else:

                print("SQL ERROR")

        conn.execute(text("COMMIT"))

    except SQLAlchemyError as e:

        print(f"SQL ERROR {e}")

    finally:

        conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

### Gemini-Pro Convert Simple COBOL Batch DB2 Program (Appendix 1.2) to Python

*Temperature:* 0.2

*Prompt:*

You are a helpful AI assistant.

You convert Cobol code into Python code. Please do not explain the code, just provide the code.

Provide a main method to run the application. Also do not omit any code for brevity. We want to see the whole code.

Hints:

\* Cobol code accesses DB2 database tables via EXEC SQL..END-EXEC statements

\* Assume equivalent Oracle table exists with similar schema as the DB2 tables

\* Access Oracle table via SQLAlchemy package in python

User input Cobol code is delimited by triple quotes. Convert the entire Cobol code to Python.

*Token Usage:*

Input tokens = 2648

*Output Python Code:*

import sqlalchemy

import cx\_Oracle

engine = sqlalchemy.create\_engine("oracle://scott:tiger@localhost:1521/orcl")

connection = engine.connect()

metadata = sqlalchemy.MetaData()

employee = sqlalchemy.Table("EMPLOYEE", metadata, autoload\_with=engine)

recordcount = connection.execute(

    sqlalchemy.select([sqlalchemy.func.count("\*")]).select\_from(employee)

).fetchone()[0]

displayrc = recordcount

out = "THE NUMBER OF EMPLOYEES IN THE DATABASE IS " + str(displayrc)

if recordcount == 0:

    out = "NO EMPLOYEES FOUND IN DATABASE"

else:

    nextemp = connection.execute(

        sqlalchemy.select([sqlalchemy.func.max(employee.c.EMPNO)]).select\_from(employee)

    ).fetchone()[0]

    nextempno = nextemp + 10

    new\_employee = employee.insert().values(

        EMPNO=nextempno,

        FIRSTNME="FRANK",

        MIDINIT="Y",

        LASTNAME="JONES",

        WORKDEPT="A00",

        PHONENO="1234",

        HIREDATE="04-30-1979",

        JOB="Clerk",

        EDLEVEL=15,

        SEX="M",

        BIRTHDATE="05-30-1954",

        SALARY=36170,

        BONUS=400,

        COMM=2387,

    )

    connection.execute(new\_employee)

    connection.commit()

    out = " ".join(["ADDED TO THE DATABASE", "FRANK", "JONES"])

print(out)

### Gemini-Pro Convert CICS MAP COBOL (Appendix 1.3) to Python

*Temperature:* 0.2

*Prompt:*

You are AI coding assistant.

For a given IBM Mainframe CICS COBOL code and BMS map, convert to Python Django MVC application.

Hint:

\* Convert BMS map to Django html template files capturing all the fields with validations

\* Convert EXEC CICS LINK statements to python method call imported from callable\_routine package

BMS map code is given below delimited by triple quotes

CICS Cobol program using the above map is given below delimited by triple quotes

"""

*Token Usage:*

Input tokens = 5620

*Output Python Code:*

from django.shortcuts import render

from django.http import HttpResponse

from django.views.decorators.csrf import csrf\_exempt

from callable\_routine import calculate\_mortgage

# Create your views here.

@csrf\_exempt

def eps\_mortgage\_calculator(request):

    if request.method == 'POST':

        loan\_amount = request.POST.get('loan\_amount')

        loan\_years = request.POST.get('loan\_years')

        interest\_rate = request.POST.get('interest\_rate')

        # Validate the input

        if not loan\_amount or not loan\_years or not interest\_rate:

            return HttpResponse('Invalid input. Please enter all fields.')

        # Calculate the monthly payment

        monthly\_payment = calculate\_mortgage(loan\_amount, loan\_years, interest\_rate)

        # Render the results page

        return render(request, 'results.html', {'monthly\_payment': monthly\_payment})

    else:

        # Render the main page

        return render(request, 'index.html')

<!-- index.html -->

<!DOCTYPE html>

<html>

<head>

    <title>EPS Mortgage Calculator</title>

</head>

<body>

    <h1>EPS Mortgage Calculator</h1>

    <form method="post">

        <label for="loan\_amount">Loan Amount:</label>

        <input type="text" name="loan\_amount" id="loan\_amount">

        <br>

        <label for="loan\_years">Loan Years:</label>

        <input type="text" name="loan\_years" id="loan\_years">

        <br>

        <label for="interest\_rate">Interest Rate:</label>

        <input type="text" name="interest\_rate" id="interest\_rate">

        <br>

        <input type="submit" value="Calculate">

    </form>

</body>

</html>

<!-- results.html -->

<!DOCTYPE html>

<html>

<head>

    <title>EPS Mortgage Calculator Results</title>

</head>

<body>

    <h1>EPS Mortgage Calculator Results</h1>

    <p>Your monthly payment is: {{ monthly\_payment }}</p>

</body>

</html>

### GPT-4 Convert CICS MAP COBOL (Appendix 1.3) to Python

*Temperature:* 0.2

*Prompt:*

You are AI coding assistant.

For a given IBM Mainframe CICS COBOL code and BMS map, convert to Python Django MVC application.

Hint:

\* Convert BMS map to Django html template files capturing all the fields with validations

\* Convert EXEC CICS LINK statements to python method call imported from callable\_routine package

*Token Usage:*

CompletionUsage(completion\_tokens=442, prompt\_tokens=1105, total\_tokens=1547)

*Output Python Code:*

"""

The conversion of the given BMS map and COBOL program to Django MVC application involves creating Django models, views, templates, and forms.

Here is a simplified version of the conversion:

Django Model:

-------------

"""

from django.db import models

class MortgageCalculator(models.Model):

    loan\_amount = models.DecimalField(max\_digits=12, decimal\_places=2)

    loan\_years = models.IntegerField()

    interest\_rate = models.DecimalField(max\_digits=5, decimal\_places=2)

    monthly\_payment = models.DecimalField(max\_digits=12, decimal\_places=2, blank=True, null=True)

"""

Django View:

-------------

"""

from django.shortcuts import render

from .models import MortgageCalculator

from .forms import MortgageCalculatorForm

def calculate\_mortgage(request):

    if request.method == 'POST':

        form = MortgageCalculatorForm(request.POST)

        if form.is\_valid():

            mortgage = form.save(commit=False)

            # Perform mortgage calculation here

            # mortgage.monthly\_payment = calculated\_value

            mortgage.save()

            return render(request, 'result.html', {'mortgage': mortgage})

    else:

        form = MortgageCalculatorForm()

    return render(request, 'index.html', {'form': form})

"""

Django Form:

-------------

"""

from django import forms

from .models import MortgageCalculator

class MortgageCalculatorForm(forms.ModelForm):

    class Meta:

        model = MortgageCalculator

        fields = ['loan\_amount', 'loan\_years', 'interest\_rate']

"""

Django Templates:

-------------

index.html:

<form method="post">

    {% csrf\_token %}

    {{ form.as\_p }}

    <button type="submit">Calculate</button>

</form>

result.html:

<p>Monthly Payment: {{ mortgage.monthly\_payment }}</p>

```

Note: This is a simplified version of the conversion. The actual conversion might require additional code to handle edge cases, error handling, and user interface improvements. The mortgage calculation logic is also not included in this conversion.

"""