ANSWER BOOK

DESIGN OF INDUSTRIAL INFORMATION SYSTEMS

Thomas O. Boucher
Ali Yalcin
CHAPTER 2 ANSWERS

2.1

a) SELECT *
   FROM PO_DETAIL
   WHERE PO_NUMBER=2594;

b) SELECT DISTINCT PO_NUMBER
   FROM PO_DETAIL
   WHERE STATUS="OPEN";

c) SELECT PO_NUMBER, PO_LINE_ITEM, MATERIAL_ID, QUANTITY, UNIT_COST, (QUANTITY*UNIT_COST) AS TOTAL_COST
   FROM PO_DETAIL
   WHERE PO_NUMBER IN
   (SELECT PO_NUMBER
    FROM PURCHASE_ORDER
    WHERE VENDOR_ID="V250");

d) SELECT PO_NUMBER, PO_LINE_ITEM, MATERIAL_ID, QUANTITY, UNIT_COST, (QUANTITY*UNIT_COST) AS TOTAL_COST
   FROM PO_DETAIL
   WHERE PO_NUMBER IN
   (SELECT PO_NUMBER
    FROM PURCHASE_ORDER
    WHERE VENDOR_ID="V250")
   ORDER BY (QUANTITY*UNIT_COST);

e) SELECT COUNT(PO_AMT)
   FROM PURCHASE_ORDER
   WHERE PO_AMT>3000;

f) SELECT DISTINCT VENDOR_ID
   FROM PURCHASE_ORDER
   WHERE PO_NUMBER IN
   (SELECT PO_NUMBER
    FROM PO_DETAIL
    WHERE (QUANTITY*UNIT_COST)>500
    AND STATUS="OPEN");

g) SELECT VENDOR.V_NAME, PO_DETAIL.PO_NUMBER, PO_DETAIL.PO_LINE_ITEM, (PO_DETAIL.QUANTITY*PO_DETAIL.UNIT_COST) AS TOTAL_COST
   FROM PURCHASE_ORDER, VENDOR, PO_DETAIL
   WHERE VENDOR.VENDOR_ID=PURCHASE_ORDER.VENDOR_ID
   AND( PURCHASE_ORDER.PO_NUMBER=PO_DETAIL.PO_NUMBER
        AND PURCHASE_ORDER.PO_AMT>2000
        AND PO_DETAIL.STATUS="OPEN");
Question 2.2 a)

BRANCH (PK: BRANCH_ID)
CLASS (PK: CLASS)
MODEL (PK: MODEL, YEAR) (FK: CLASS)
VEHICLE (PK: VIN) (FK: BRANCH_ID, MODEL, YEAR)

Question 2.2 b)  Database available to instructors from Web site.

Question 2.2 c)

1) SELECT VIN
FROM CAR
WHERE MODEL_ID IN
(SELECT MODEL_ID
FROM MODEL
WHERE MAKE="Chevrolet");

2) SELECT CAR.VIN, MODEL.YEAR
FROM CAR, MODEL, LOCATION
WHERE MODEL.MAKE="Chevrolet" AND (MODEL.MODEL="Impala" AND
MODEL.MODEL_ID=CAR.MODEL_ID AND LOCATION.CITY="St. Petersburg" AND
LOCATION.LOCATION_ID=CAR.LOCATION_ID);

3) SELECT LOCATION.*, CAR.VIN
FROM LOCATION, CAR, MODEL
WHERE LOCATION.LOCATION_ID=CAR.LOCATION_ID AND (MODEL.MODEL="Passat"
AND MODEL.MODEL_ID=CAR.MODEL_ID);

4) SELECT LOCATION_ID, VIN
FROM CAR
WHERE COLOR="Red";

5) SELECT COUNT(MODEL_ID)
FROM CAR
WHERE MODEL_ID IN
(SELECT MODEL_ID
FROM MODEL
WHERE CLASS = "Full Size");

6) SELECT COUNT(CAR.MODEL_ID)
FROM CAR, MODEL, LOCATION
WHERE CAR.MODEL_ID=MODEL.MODEL_ID And
(CAR.LOCATION_ID=LOCATION.LOCATION_ID And MODEL.CLASS="Compact" And
LOCATION.CITY="Tampa");

7) SELECT CLASS.CLASS_ID, (7*DAILY_RATE-WEEKLY_RATE) AS [Cost Saving]
FROM CLASS;

8) SELECT DISTINCT CAR.LOCATION_ID, MODEL.CLASS
FROM CAR, MODEL
WHERE CAR.MODEL_ID=MODEL.MODEL_ID;
2.3

Question 2.3 a)

EMPLOYEE (PK: EMPL_ID) (FK: DEPT_ID)
SKILL (PK: SKILL_ID)
DEPARTMENT (PK: DEPT_ID)
EMPLOYEE_SKILL_SET (PK, FK: EMPL_ID, SKILL_ID)

Question 2.3 b)

1) SELECT FIRST_NAME, LAST_NAME
FROM EMPLOYEE
WHERE DEPT_ID="30";

2) SELECT FIRST_NAME, LAST_NAME
FROM EMPLOYEE
WHERE EMPL_ID IN
  ( SELECT EMPL_ID FROM EMPLOYEE_SKILL_SET
    WHERE SKILL_ID="110");

3) SELECT EMPLOYEE.FIRST_NAME, EMPLOYEE.LAST_NAME
FROM EMPLOYEE, SKILL, EMPLOYEE_SKILL_SET
WHERE EMPLOYEE.EMPL_ID=EMPLOYEE_SKILL_SET.EMPL_ID
AND SKILL.SKILL_DESC="C Programmer"
AND SKILL.SKILL_ID=EMPLOYEE_SKILL_SET.SKILL_ID;

4) SELECT DISTINCT COUNT(EMPL_ID)
FROM EMPLOYEE_SKILL_SET
WHERE EMPLOYEE_SKILL_SET.SKILL_ID IN
  (SELECT SKILL_ID
   FROM SKILL
   WHERE SKILL_DESC="Java Programmer");

5) SELECT EMPLOYEE.LAST_NAME, EMPLOYEE.FIRST_NAME
FROM EMPLOYEE, SKILL, EMPLOYEE_SKILL_SET
WHERE EMPLOYEE.DEPT_ID="30"
AND EMPLOYEE_SKILL_SET.EMPL_ID=EMPLOYEE.EMPL_ID
AND SKILL.SKILL_ID=EMPLOYEE_SKILL_SET.SKILL_ID
AND SKILL.SKILL_DESC="Java Programmer"
ORDER BY EMPLOYEE.LAST_NAME ASC;

6) SELECT DISTINCT EMPLOYEE.DEPT_ID, DEPARTMENT.DEPT_NAME
FROM EMPLOYEE, DEPARTMENT, EMPLOYEE_SKILL_SET, SKILL
WHERE EMPLOYEE.DEPT_ID=DEPARTMENT.DEPT_ID
AND EMPLOYEE_SKILL_SET.EMPL_ID=EMPLOYEE.EMPL_ID
AND SKILL.SKILL_ID=EMPLOYEE_SKILL_SET.SKILL_ID
AND NOT SKILL.SKILL_DESC = "Java Programmer";
2.4

Question 2.4 a)
Database available on Web site.

Question 2.4 b)

1) SELECT *
   FROM PART;

2) SELECT *
   FROM PART
   WHERE MATERIAL_ID = "RM302";

3) SELECT *
   FROM PART
   ORDER BY PART.PART_DESC;

4) SELECT *
   FROM SCHEDULE
   WHERE S_LOT_SIZE > 15;

5) SELECT *
   FROM PART
   WHERE PROCESS_PLAN_ID IN (SELECT PROCESS_PLAN_ID
                               FROM PROCESS_PLAN
                               WHERE MACHINE_ID = "M1");

6) SELECT DISTINCT PROCESS_PLAN.MACHINE_ID,
    AVG(PROCESS_PLAN.STD_UNIT_MACH_MIN),
    MIN(PROCESS_PLAN.STD_UNIT_MACH_MIN
    MAX(PROCESS_PLAN.STD_UNIT_MACH_MIN)
    FROM PROCESS_PLAN
    GROUP BY PROCESS_PLAN.MACHINE_ID;

7) SELECT SCHEDULE.SCHEDULE_ID, PART.PART_NO,
    PROCESS_PLAN.OP_NO, PROCESS_PLAN.MACHINE_ID,
    PROCESS_PLAN.STD_SU_MIN,
    PROCESS_PLAN.STD_UNIT_MACH_MIN*SCHEDULE.S_LOT_SIZE
    AS TOT_MACH_MIN,
    PROCESS_PLAN.STD_SU_MIN+
    (PROCESS_PLAN.STD_UNIT_MACH_MIN*SCHEDULE.S_LOT_SIZE) AS TOT_MIN
    FROM SCHEDULE, PART, PROCESS_PLAN
    WHERE PART.PART_NO=SCHEDULE.PART_NO
     AND PART.PROCESS_PLAN_ID=PROCESS_PLAN.PROCESS_PLAN_ID;

8) SELECT DISTINCT PART.MATERIAL_ID, PROCESS_PLAN.TOOLING_ID
    FROM PART, PROCESS_PLAN
    WHERE PART.PROCESS_PLAN_ID=PROCESS_PLAN.PROCESS_PLAN_ID;
9) SELECT SCHEDULE.S_RELEASE_DATE, PART.PART_NO,
    PROCESS_PLAN.MACHINE_ID
FROM PART, PROCESS_PLAN, SCHEDULE
WHERE PART.PART_NO=SCHEDULE.PART_NO
   And PART.PROCESS_PLAN_ID=PROCESS_PLAN.PROCESS_PLAN_ID
ORDER BY SCHEDULE.S_RELEASE_DATE DESC;

10) SELECT PART.PART_NO, PROCESS_PLAN.MACHINE_ID
    FROM PART, PROCESS_PLAN, SCHEDULE
    WHERE PART.PART_NO=SCHEDULE.PART_NO
    And SCHEDULE.S_RELEASE_DATE=#4/1/2002#
    And PART.PROCESS_PLAN_ID=PROCESS_PLAN.PROCESS_PLAN_ID;

11) SELECT DISTINCT PROCESS_PLAN.MACHINE_ID
    FROM PART, PROCESS_PLAN, SCHEDULE
    WHERE PART.PART_NO=SCHEDULE.PART_NO
    And PART.PROCESS_PLAN_ID=PROCESS_PLAN.PROCESS_PLAN_ID
    And S_RELEASE_DATE=#4/1/2006#;

12) SELECT PART.PART_NO
    FROM PART
    WHERE PROCESS_PLAN_ID IN (SELECT PROCESS_PLAN_ID
                                FROM PROCESS_PLAN
                                WHERE STD_SU_MIN >8);

13) SELECT DISTINCT PART.MATERIAL_ID, SCHEDULE.S_LOT_SIZE
    FROM PART, SCHEDULE
    WHERE PART.PART_NO=SCHEDULE.PART_NO
    And SCHEDULE.S_RELEASE_DATE=#4/1/2006#;

2.5

a) SELECT *
    FROM INVOICE_LINE
    WHERE INVOICE_NO=1017
    ORDER BY PRODUCT_ID DESC;

b) SELECT INVOICE.INVOICE_NO, INVOICE.SALE_DATE,
    CUSTOMER.LAST_NAME, CUSTOMER.STATE
    FROM INVOICE, CUSTOMER
    WHERE CUSTOMER.LAST_NAME="Harris" AND
    INVOICE.CUSTOMER_NO=CUSTOMER.CUSTOMER_NO;

c) SELECT *
    FROM CUSTOMER
    WHERE LAST_NAME LIKE "Smith*";
d)  INSERT INTO PRODUCT
VALUES ("B101", "Custom Mountain Bike", 225.00);

e)  SELECT INVOICE_LINE.INVOICE_NO, INVOICE.SALE_DATE,
    PRODUCT.PRODUCT_DESC,
    PRODUCT.PRICE AS Unit_Price, INVOICE_LINE.QTY_SOLD,
    PRODUCT.PRICE*INVOICE_LINE.QTY_SOLD AS Total_Amt
FROM INVOICE_LINE, INVOICE, PRODUCT
WHERE INVOICE_LINE.INVOICE_NO=INVOICE.INVOICE_NO
AND INVOICE_LINE.PRODUCT_ID=PRODUCT.PRODUCT_ID
ORDER BY INVOICE_LINE.INVOICE_NO;

f)  SELECT CUSTOMER_NO
FROM INVOICE
WHERE INVOICE_NO IN
  (SELECT INVOICE_NO
   FROM INVOICE_LINE
   WHERE PRODUCT_ID="B1")

2.6

Question 2.6a)

CELL (PK: CELL_ID)
MACHINE (PK: MACHINE_ID)
MACHINE_CELL_XREF (PK: CELL_ID, MACHINE_ID) (FK: CELL_ID, MACHINE_ID)
PART (PK: PART_ID)
PROCESS_PLAN (PK: PROCESS_PLAN_ID) (FK: PART_ID)
PROCESS_PLAN_STEP (PROCESS_PLAN_ID, OPERATION_NO) (FK: PROCESS_PLAN_ID,
                      MACHINE_ID)

Question 2.6b)

1)  SELECT MACHINE_CELL_XREF.CELL_ID, MACHINE.MACHINE_ID,
    MACHINE.MACHINE_NAME
    FROM MACHINE_CELL_XREF, MACHINE
    WHERE MACHINE_CELL_XREF.MACHINE_ID=MACHINE.MACHINE_ID
    ORDER BY MACHINE_CELL_XREF.CELL_ID, MACHINE.MACHINE_ID;

2)  SELECT [MACHINE_ID], Min([MACHINE_TIME]) AS [MIN], Max(MACHINE_TIME) AS [MAX],
    Avg(MACHINE_TIME) AS [AVG]
    FROM PROCESS_PLAN_STEP
    GROUP BY [MACHINE_ID];

3)  SELECT DISTINCT MACHINE_CELL_XREF.CELL_ID, PROCESS_PLAN.PART_ID,
    Sum(PROCESS_PLAN_STEP.MACHINE_TIME) AS TOT
    FROM MACHINE_CELL_XREF, PROCESS_PLAN, PROCESS_PLAN_STEP
    WHERE MACHINE_CELL_XREF.MACHINE_ID=PROCESS_PLAN_STEP.MACHINE_ID AND
      PROCESS_PLAN.PROCESS_PLAN_ID=PROCESS_PLAN_STEP.PROCESS_PLAN_ID
    GROUP BY CELL_ID, PROCESS_PLAN.PART_ID;
4) SELECT DISTINCT MACHINE_CELL_XREF.CELL_ID
FROM MACHINE_CELL_XREF
WHERE MACHINE_ID IN
(SELECT PROCESS_PLAN_STEP.MACHINE_ID
FROM PROCESS_PLAN_STEP
WHERE PROCESS_PLAN_STEP.MACHINE_TIME > 300);

OR,

SELECT DISTINCT MACHINE_CELL_XREF.CELL_ID
FROM MACHINE_CELL_XREF, PROCESS_PLAN_STEP
WHERE MACHINE_CELL_XREF.MACHINE_ID=PROCESS_PLAN_STEP.MACHINE_ID
AND PROCESS_PLAN_STEP.MACHINE_TIME>300;

2.7

a) SELECT QA_TEST.TEST_NO, QA_TEST.LAB_PROC_SPEC_ID, QA_TEST.TEST_RESULT AS RESULT
FROM QA_TEST
ORDER BY LAB_PROC_SPEC_ID, TEST_NO;

b) SELECT QA_TEST.TEST_NO, LAB_PROCEDURE_SPEC.LAB_PROC_SPEC_ID,
LAB_PROCEDURE_SPEC.TEST_ACCEPT_VALUE_RANGE,
QA_TEST.ACTUAL_VALUE, QA_TEST.TEST_RESULT
FROM QA_TEST, LAB_PROCEDURE_SPEC
WHERE LAB_PROCEDURE_SPEC.[LAB_PROC_SPEC_ID]=[QA_TEST].[LAB_PROC_SPEC_ID]
ORDER BY LAB_PROCEDURE_SPEC.LAB_PROC_SPEC_ID, QA_TEST.TEST_NO;

c) SELECT COUNT(TEST_KIT_ID)
FROM USAGE_TRANSACTION
WHERE TRANS_TYPE="W" AND TEST_KIT_ID IN (SELECT TEST_KIT_ID
FROM TEST_KIT
WHERE CABINET_NO=5);

OR,

SELECT COUNT(TRANS_TYPE)
FROM USAGE_TRANSACTION
WHERE TRANS_TYPE="W"
AND TEST_KIT_ID IN (SELECT TEST_KIT_ID
FROM TEST_KIT
WHERE CABINET_NO=5);

d) SELECT Min(TEST_DATE) AS [MIN], Max(TEST_DATE) AS [MAX]
FROM USAGE_TRANSACTION
WHERE USAGE_TRANSACTION.TEST_KIT_ID="TK1"
AND USAGE_TRANSACTION.TRANS_TYPE="W";
The ternary relationship could be broken into three 1:M relationships as follows:

The quality test is an identifier for a test procedure having several steps. The first step is taking the sample. Two quality tests may differ only by the sample size. At any point in time the quality test for a specific vendor and material combination is given in the cross reference table. Each combination is unique. A 1:M relationship is established among entities.
PART is made up of COMPONENT

C is made up of R2, R3, R4, R5.
3.5

a)  

MODEL: Make, Year, Model_Name, Model_ID (PK), Class_ID (FK)
CLASS: Daily_Rate, Weekly_Rate, Class_ID (PK)
CAR: VIN (PK), Branch_ID (FK), Model (FK)
LOCATION: Branch_ID (PK), Branch_Location
RESERVATION: Reservation_ID (PK), Customer_Name, Customer_Address, Class_ID (FK), Reservation_Date, Reservation_Time, Reservation_Return_Date, Reservation_Return_Time, Branch_ID
RENTAL AGREEMENT: Contract_No (PK), Cust_Operator_Lic, State, Credit_Card_Type, Credit_Card_No, Expiration_Date, VIN (FK), Date_Out, Time_Out, Odometer_Out, Date_Return, Time_Return, Odometer_In, Reservation_ID (FK)
3.6

a)

CUSTOMER: CUST_ID (PK), CUST_NAME, CUST_ADDRESS
STYLE: STYLE_ID (PK), STYLE_DESCRIPTION
COLOR: COLOR_ID (PK), COLOR_DESC
STORE: (CUST_ID, STORE_ID) (PK), STORE_ADDRESS
SKU: UPC (PK), STYLE_ID (FK), COLOR_ID (FK), WAIST, LENGTH
STORE_SKU_XREF: (STORE_ID, UPC) (PK), STD_SALES, STD_RETURNS, ON_HAND, IN_TRANSIT, ON_ORDER, PERPETUAL_INV.

b)

CUSTOMER: CUST_ID (PK), CUST_NAME, CUST_ADDRESS
STYLE: STYLE_ID (PK), STYLE_DESCRIPTION
COLOR: COLOR_ID (PK), COLOR_DESC
STORE: (CUST_ID, STORE_ID) (PK), STORE_ADDRESS
SKU: UPC (PK), STYLE_ID (FK), COLOR_ID (FK), WAIST, LENGTH
STORE_SKU_XREF: (STORE_ID, UPC) (PK), STD_SALES, STD_RETURNS, ON_HAND, IN_TRANSIT, ON_ORDER, PERPETUAL_INV.
3.8 University Food Case (A)

a) 

b) 11.

c) Recipe: Recipe_ID (PK), Recipe_Name, Recipe_Description, Revision_Date

Production_Step: Production_Step_ID (PK), Recipe_ID (PK, FK), PS_Description.

Formula: Formula_ID (PK), Recipe_ID (FK), Formula_Description, Formula_Quantity.

Formula_Detail: Formula_ID (PK, FK), Ingredient_ID (PK, FK), Quantity, Unit_of_Measure.
3.9 ACME Case (A).

Note: 1) The relationship between routing sheet and operation may be M:N if the same operation appears on more than one routing sheet.
2) Employee can be replaced with operator. Employee or operator does not have a direct association with equipment unless skill class is shown as an attribute of equipment. However, this association is acceptable and the attribute (not shown) will be understood.
3.

1) Note that you have to add an attribute not in evidence to obtain 1\textsuperscript{st} NF.

Routing Sheet (\text{Routing Sheet No}, \text{Part No}, \text{Part Name}, \text{Qty}, \text{Mat’l Name}, \text{Operation No}, \text{Description}, \text{Eqpt}, \text{Tooling})

2) Routing Sheet (\text{Routing Sheet No}, \text{Part No}, \text{Part Name}, \text{Qty}, \text{Material Name})
Routing Operation (\text{Routing Sheet No}, \text{Operation No}, \text{Desc}, \text{Eqpt}, \text{Tooling})

3) Part (\text{Part No}, \text{Part Name}, \text{Mat’l Name})
Routing Sheet (\text{Routing Sheet No}, \text{Part No}, \text{Qty})

Note the limitations of 3\textsuperscript{rd} NF normalization if you do not have Mat’l ID and Eqpt ID given in the table.
3.10 Regal Foreign Car Repairs Case (A)

a)

![ER Diagram]

Note: Since each repair order has only one cost summary, statement 8 can be said to just describe attributes of the entity Repair Order. On the other hand, it is not incorrect to include it in the E-R model with a 1:1 relationship to Repair Order.

b)

CUSTOMER (Customer ID (PK), Customer Name, Mailing Address, Telephone No.)

AUTOMOBILE (Registration no. (PK), Chassis no., Engine no., Make, Model, Year, Customer_ID (FK))
REPAIR_ORDER (Repair Order no. (PK), Registration no. (FK), Milage, Date & Time In, Date & Time Out, Comments, Labor Costs, Part Costs, Total Cost, Payment Method, Credit Card Type, Credit Card no.)

RO_DETAIL ([Repair Order no.; Item no.](PK), Trouble Code (FK), Flat Hrs., Mechanic ID (FK) Act. Hrs.)

TROUBLE_CODE (Trouble Code ID (PK), TC Description)

MECHANIC (Mechanic ID (PK), Mechanic Name, Labor Rate)

PART (Part ID (PK), Part Description, Part Cost*)

PART_DETAIL ([Repair Order No, Item No, Part_Detail_No] (PK), Part_ID (FK), Quantity)

* Here we are assuming that the cost of a part, which is updated in the PART table whenever it changes, will not necessarily be consistent with the summary “Part Costs” in the REPAIR_ORDER table. In order to maintain a consistent history it is necessary to introduce time intervals for when a part cost was relevant and to associate that history with the REPAIR_ORDER table. This will be taken up in a later chapter.
4.1 Controls give a picture of how a system actually operates. The classic illustration concerns the assembly of a chair. The chair legs, back, and seat are clearly inputs to the assembly process. However, they do not describe how to do the assembly. For this you need a blueprint and/or a set of instructions. In the case of a physical transformation process, like the assembly of a chair, it is easy to see the difference between an input and a control.

When it comes to information, it is less clear whether it is an input or a control. The production schedule and recipe describe what to produce on a specific day as well as the steps involved in the process. The production schedule gives direction to the worker about what to do on a specific product day and the recipe gives the step-by-step procedure of how to make the product. Therefore, they are properly characterized as a control. The material move schedule is less conclusive. It does provide direction to the worker as to which material lots to draw from inventory and where to put them. However, it does not tell him the steps to use in doing these tasks. This is a case where some judgment is involved. We conclude it is more of a control than an input because it provides specific instruction about which lots to move and it is the only guide to follow when performing the function.

4.2
1.0 Confirm Validity of Shipment

2.0 Inspect Condition of Material

3.0 Receive Materials

Purchasing File

QA Files

Inventory File

Warehouse File

Transporter or Vendor

Bill of Laded

Purchase Order

Notification to Inspect

Inspection Criteria

Sequential Lot Numbers

Available Locations

Acceptance Data, Material Lot Number Assignments

Receiving Report File

Storage Processes
4.4

CUSTOMER

Request to pick up car

Copy of rental agreement

Service Customer at Rental Pick Up

CUSTOMER

Reservation file

Reservation inquiry

1.0 Check for reservation

2.0 Make reservation

3.0 Draw up rental agreement

Reservation information data

Reservation Information data

Rental Information

Copy of rental agreement

Car file

Car availability

Car assignment

Rental agreement file
4.5 a)
4.5 b)

Manufacturing

Daily hours of operation

1.0 Update machine operating hours

Cumulative machine operating hours

Machine operating hours file

2.0 Monitor hours of operation

Cumulative hours of operation

Maintenance records

3.0 Perform preventive maintenance

Date last performed

Work order

Open work order file

Completed work order file

Completed work order

4.0 Date and transfer records

Maintenance schedule file

Task schedule

Maintenance manual

Maintenance instructions

Completed work order

Closed work order
4.6 IDEF0
DATA FLOW DIAGRAM:

CONTEXT DIAGRAM

LEVEL 1 DIAGRAM
5.1

There are two candidate answers shown below. We prefer candidate (a). These two approaches can serve the purpose of class discussion when students hand in their assignments. The merits of the two approaches can be discussed, thus showing that more than one model is possible.

a) In this model the purchasing document data are separated from the material lot inventory document data by the entity DELIVERY_TRANSACTION. The delivery transaction mirrors the receiving report information by providing a record of the receiving of a line item on a purchase order on a particular delivery. The entity DELIVERY carries the general information about the delivery and DELIVERY_TRANSACTION accounts for the handling of each line item during the delivery. The quantity delivered on a line item may be divided into more than one lot depending of the number of different vendor lot numbers in the delivered quantity. Thus, a one to many relationship exists between DELIVERY_TRANSACTION and MATERIAL_LOT. Also, DELIVERY_TRANSACTION accounts for the amount of the line item that is returned.
b) This solution will also work, but it is not as good as (a). Here the PO_DETAIL is associated with the material lot instead of the delivery transaction. The delivery information is still provided in the entity DELIVERY. It is still possible to divide a delivered line item into more than one lot number. An entity RETURN is required to keep track of those quantities on a delivery that were returned.
5.2 When LOCATION_TYPE is added.

5.3
5.6

5.7 ACME MAINTENANCE:
If you keep the total machine hours once, which all that is required.

If you keep the total hours at the end of each day:
SQL CODE:

SELECT MACHINE.MACHINE_NAME, MACHINE.MACHINE_ID AS MACHINE_SERIAL_NO,
    MACHINE.INSTALL_DATE AS FIRST_DATE_IN_SERVICE,
    MACHINE.WARRENTY_EXP_DATE
FROM MACHINE;

SELECT MACHINE.MACHINE_NAME, PM_WORK_ORDER.PM_TASK_ID,
    PM_WORK_ORDER.DATE_COMPLETED, PM_WORK_ORDER.MACHINE_HOURS, PM_WORK_ORDER.MINUTES_WORKED
FROM MACHINE, PM_WORK_ORDER
WHERE MACHINE.MACHINE_ID=PM_WORK_ORDER.MACHINE_ID
ORDER BY MACHINE.MACHINE_NAME, PM_WORK_ORDER.PM_TASK_ID;
Note: In the above model orders may be split among transfers and shipments. A transfer is a combination of an ORDER_NO, LINE_ITEM and LOT_NO. Therefore, the same order number and line item can be satisfied by more than one lot number. Also, the shipping number is the same as the bill of laden number. So the bill of laden can be reconstructed from information in the following entities: SHIPMENT, TRANSFER, C_ORDER_DETAIL, MATERIAL, C_ORDER, CUSTOMER.

SQL Code

SELECT [C_ORDER].[ORDER_NO], [C_ORDER].[CUSTOMER_ID], [CUSTOMER].[C_NAME], [CUSTOMER].[C_STREET], [CUSTOMER].[C_CITY], [CUSTOMER].[C_STATE], [C_ORDER].[PO_NUMBER], [C_ORDER_DETAIL].[MATERIAL_ID], [MATERIAL].[MATL_DESCRIPTION], [C_ORDER_DETAIL].[ORDER_QTY], [C_ORDER_DETAIL].[UNIT_PRICE], [C_ORDER_DETAIL].[UNIT_PRICE]*[C_ORDER_DETAIL].[ORDER_QTY] AS EXTENSION, [C_ORDER_DETAIL].[PROMISED_DEL_DATE], FROM C_ORDER, CUSTOMER, C_ORDER_DETAIL, MATERIAL
WHERE [C_ORDER].[CUSTOMER_ID]=[CUSTOMER].[CUSTOMER_ID] And [C_ORDER].[ORDER_NO]=[C_ORDER_DETAIL].[ORDER_NO] And [C_ORDER_DETAIL].[MATERIAL_ID]=[MATERIAL].[MATERIAL_ID] And [C_ORDER_DETAIL].[ORDER_BALANCE]>0
ORDER BY C_ORDER_DETAIL.PROMISED_DEL_DATE, C_ORDER.ORDER_NO;
Note that the part cost is now allowed to be revised and the revision number is carried in the PART_DETAIL table. The Active_Bit is an indicator of which revision number is currently active (1). All inactive revisions have Active_Bit = 0. There are other possible solutions, but this one should suffice.
CHAPTER 6 ANSWERS

6.1. 

<table>
<thead>
<tr>
<th>Control Stored Mat'l</th>
<th>Material Lot</th>
<th>Warehouse Location</th>
<th>Lot Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store Raw Materials</td>
<td></td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Move RM to WIP</td>
<td>U</td>
<td>U</td>
<td>I</td>
</tr>
<tr>
<td>Return RM to Storage</td>
<td>U</td>
<td>U</td>
<td>I</td>
</tr>
</tbody>
</table>

6.2. Answers on instructor database, Material_manager_CH6_instructor.

6.3. Answers on instructor database, Material_manager_CH6_instructor.

6.4. Answers on instructor database, Material_manager_CH6_instructor.

6.5. There are many ways to implement these data integrity requirements. For Vendor ID, one could use the IN operator as follows: IN (“V25”, “V75”, “V110”, “V250”). The material ID has many entries and a preferred way to do this might be to search the material table for the existence of the material ID before accepting the entry. This can be done using DAO and the techniques shown in the password example, Figure 6.18. It might also be simpler to add combo boxes to choose the vendor ID and material ID. This guarantees that they exist in the vendor and material tables and the user only has to enter the v item code. Finally, if you lock the subforms, no integrity check is needed for unit of measure.

6.6 Answers on instructor database, Material_manager_CH6_instructor.

6.7 Answers on instructor database, Material_manager_CH6_instructor.

6.8 Answers on instructor database, Material_manager_CH6_instructor.

6.9 Answers on instructor database, Material_manager_CH6_instructor.

6.10 Both reports are implemented in the database titled ACME_Maintenance_instructor. Machine Warranty Form is based on the table MACHINE. PM Maintenance History Form is based on the Query: PM Maintenance History Query as follows:

```
SELECT MACHINE.MACHINE_NAME, PM_WORK_ORDER.PM_TASK_ID,  
PM_WORK_ORDER.DATE_COMPLETED, PM_WORK_ORDER.MACHINE_HOURS,  
PM_WORK_ORDER.MINUTES_WORKED  
FROM MACHINE, PM_WORK_ORDER  
WHERE (((MACHINE.MACHINE_ID)=[PM_WORK_ORDER].[MACHINE_ID]));
```

6.11 University Food Receiving Department – Case (D)

The completed database is UniversityFoodReceiving_CH6_Instructor.

a) The process of developing this form includes the following:
1. The Master Form is based on the PURCHASE_ORDER Table. Form Properties: Scroll Bar – neither; Record Selector – No.
2. The PO_Detail Subform Properties: Scroll Bar – neither; Record Selector – No. Link Master Fields and Child Fields: PO_NUMBER.
3. Vendor Subform pProperties: Scroll Bar – Neither; Record Selector – No; Navigation Buttons – No. Link Master and Child Fields: VENDOR_ID.
4. Vendor_Matl_Xref Subform Properties: Scroll Bar – Neither; Record Selector – No. Link Master and Child Fields: VENDOR_ID.
5. Material_Lot Subform Properties: Scroll Bar – Neither; Record Selector – No. Link Master and Child Fields: PO_NUMBER.

Lock all text boxes for read-only except text boxes for entering data on the Material Lot Subform.

b) The process of developing this form is as follows:

1. The Warehouse Locator Master Form is based on the table MATERIAL_LOT. Format Properties: Scroll Bar – Neither; Record Selector – No.
2. The Warehouse Location Subform is based on the WAREHOUSE_LOCATION table and uses the datasheet view. Format Properties: Record Selection – No; Navigation Buttons – No. Link Master and Child Fields: MATL_LOT_NO.
3. The Data Entry Subform is also based on the WAREHOUSE_LOCATION table. This form should not be linked to the master. Remove Link Master and Link Child Fields that may be automatically provided by Access.

Lock all text boxes except the data entry subform text boxes.

6.12 ACME Machine Shop – Case (D):

b) There are several steps in the creation of the form for updating machine operating hours. The following steps are one way to solve the problem.
1. Create both the main form and the subform.

**Access Exercise to create main form layout:**

1. Open the ACME_MAINTENANCE database.
2. Open the **Relationships** window and create the relationships as shown below.
3. Go to the **Forms** menu and select “Create form by using wizard”.
4. In the Table/Query textbox choose the table **MACHINE**.
5. Select the fields: MACHINE_ID, MACHINE_NAME, MANUFACTURER_ID, and MACHINE_TOTAL_HOURS. Click on Next.
7. Select Standard style. Click Next.
8. Use the name of the function for which the form is to be used as the title of the form: **UPDATE_MACHINE_HOURS**. Click on Finished.
9. Put the form in design view and alter the appearance of the form until it looks like Figures 2 and 3. Ignore the “Update” command button for the time being.
10. Set the locked property of the text boxes to yes.
11. Save and close the form.

**Access Exercise to create Sub form layout:**

1. Go to the **Forms** menu and select “Create form by using wizard”.
2. In the Table/Query textbox choose the table **MACHINE_USE_RECORD**.
3. Select all the available fields. Click on Next.
4. Select Columnar Layout. Click Next.
5. Select Standard style. Click Next.
6. Entitle the form: **UPDATE_MACHINE_HOURS_SUBFORM**. Click on Finished.
7. Put the form in design view and alter the appearance of the form until it looks like Figures 2 and 3. You will have to eliminate the scroll bars.
8. Eliminate the scroll bars: Open the properties window of the form. Under the Format properties, set the following: Scroll Bars => Neither; Record Selectors => No; Navigation Buttons => No.
9. Set the locked property of the text boxes to yes.
10. Save and close the form.
2. Add the command button to the subform.

The command button on the subform is an “Insert Record” button. The code for the click() event procedure of an insert record button is one line: DoCmd.GoToRecord , , acNewRec. This command simply tells the DBMS to go to a new record position of the dynaset on which the form is based; position n+1, where n is the current number of existing records. When the user enters data in the new record fields, the insertion of the record is complete. There are additional requirements given in the problem concerning what is to happen on click. The additional requirements are to unlock the text boxes and to set the focus (cursor) in the DATE textbox. In order to do this, the following code is required in the click() event procedure:

```
DoCmd.GoToRecord , , acNewRec
Me.DATE.Locked = False
Me.HOURS_OF_OPERATION.Locked = False
Me.DATE.SetFocus
```

Access exercise to add the Insert Record command button to the subform.

1) Make enough space available on the Detail section of the subform.
2) From the Toolbox, select and place a command button on the subform. The command button wizard window will appear.
3) Select “Record Operations” and “Add New Record”. Click on Next.
4) Select Picture: GoTo New2. Click on Next.
5) Name the button: Insert_Button. Click on Finished.
6) Open the Insert_Button properties window and open the On Click Event Procedure.
7) After the existing DoCmd code, add the three additional lines of code as shown above. Close the properties window.
8) Add a label above the Insert_Button as shown in Figure 3.
3. Bind the subform to the main form.

Access exercise to bind subform to main form.

1) With the main form open in design view and the subform closed, go to the database window, forms objects, and select the subform.
2) While holding down the left mouse button, drag the subform to the appropriate position on the main form. Release the button.
3) Delete the label of the subform.
4) Open the subform properties window and, under Data properties, enter the attribute MACHINE_ID in “Link Child Fields” and “Link Master Fields”.

4. Add the update command button to the main form

As described in the assignment, the function of the “Update” command button is to 1) Unlock MACHINE_TOTAL_HOURS textbox, 2) Add the contents of the HOURS_OF_OPERATION textbox from the subform to the MACHINE_TOTAL_HOURS textbox of the main form and 3) Lock the DATE, HOURS_OF_OPERATION, and MACHINE_TOTAL_HOURS textboxes. The Event Procedure code to perform these operations is as follows:

Private Sub Update_Button_Click()
    Me.MACHINE_TOTAL_HOURS.Locked = False
    Me.MACHINE_TOTAL_HOURS.Value = MACHINE_TOTAL_HOURS.Value + [UPDATE_MACHINE_HOURS_SUBFORM]!HOURS_OF_OPERATION.Value
    Me.MACHINE_TOTAL_HOURS.Locked = True

    [UPDATE_MACHINE_HOURS_SUBFORM]!DATE.Locked = True
    [UPDATE_MACHINE_HOURS_SUBFORM]!HOURS_OF_OPERATION.Locked = True
End Sub

Access exercise to add the Update Record command button to the main form.

1) Make enough space available on the Detail section of the subform.
2) From the Toolbox, select and place a command button on the main form. The command button wizard window will appear.
3) Select “Record Operations” and “Add New Record”. Click on Next.
4) Select Picture: GoTo New2. Click on Next.
5) Name the button: Insert_Button. Click on Finished.
6) Open the Insert_Button properties window and open the On Click Event Procedure.
7) After the existing DoCmd code, add the three additional lines of code as shown above. Close the properties window.
8) Add a label above the Insert_Button as shown in Figure 3.
c) There are several steps in the creation of the form for monitoring hours of operation. The following steps are one way to solve the problem.

1. Create both the main form and the subform.

   **Access Exercise to create main form layout:**

   12) Open the ACME MAINTENANCE database.
   13) Open the Relationships window and create the relationships as shown below.
   14) Go to the Forms menu and select “Create form by using wizard”.
   15) In the Table/Query textbox choose the table MACHINE.
   16) Select the fields: MACHINE_ID, MACHINE_NAME, MANUFACTURER_ID, and MACHINE_TOTAL_HOURS. Click on Next.
   17) Select Columnar Layout. Click Next.
   18) Select Standard style. Click Next.
   19) Use the name of the function for which the form is to be used as the title of the form: UPDATE_MACHINE_HOURS. Click on Finished.
   20) Put the form in design view and alter the appearance of the form until it looks like Figures 2 and 3. Ignore the “Update” command button for the time being.
   21) Set the locked property of the text boxes to yes.
   22) Save and close the form.

The subform will be based on a query. Before designing the form, choose the Query option on the database window and enter the following query:

**MONITOR_HOURS_QUERY:**

SELECT [PM_TASK].[PM_TASK_ID], [MACHINE_PM_TASK_XREF].[MACHINE_ID],
       [MACHINE_PM_TASK_XREF].[MACHINE_PM_TASK_XREF].[MACH_HRS_LAST_DONE],
       [PM_TASK].[PM_TASK_FREQUENCY], [MACHINE].[MACHINE_TOTAL_HOURS]
FROM PM_TASK, MACHINE_PM_TASK_XREF, MACHINE
WHERE [MACHINE_PM_TASK_XREF].[PM_TASK_ID]=[PM_TASK].[PM_TASK_ID] And
       [MACHINE_PM_TASK_XREF].[MACHINE_ID]=[MACHINE].[MACHINE_ID];

   **Access Exercise to create Sub form layout:**

   11) Go to the Forms menu and select “Create form by using wizard”.
   12) In the Table/Query textbox choose the query MONITOR_HOURS_QUERY.
   13) Select all the available fields. Click on Next.
   14) Select Columnar Layout. Click Next.
   15) Select Standard style. Click Next.
   16) Entitle the form: MONITOR_HOURS_SUBFORM. Click on Finished.
   17) Put the form in design view and alter the appearance of the form until it looks like Figures 2 and 3. You will have to eliminate the scroll bars.
   18) Eliminate the scroll bars: Open the properties window of the subform. Under the Format properties, set the following: Scroll Bars => Neither; Record Selectors => No; Navigation Buttons => No.
   19) Set the locked property of the text boxes to yes.
   20) Save and close the form.
2. Add the derived attribute to the subform.

In design view, open the toolbox, select a textbox and place it on the Detail section of the subform. Change the label to the caption: ELAPSED HRS. place the cursor into the textbox and type the following arithmetic expression: \([\text{MACHINE\_TOTAL\_HOURS}] – [\text{MACH\_HRS\_LAST\_DONE}]\). Save and close the subform.

3. Bind the subform to the main form.

Access exercise to bind subform to main form.

5) With the main form open in design view and the subform closed, go to the database window, forms objects, and select the subform.

6) While holding down the left mouse button, drag the subform to the appropriate position on the main form. Release the button.

7) Delete the label of the subform.

8) Open the subform properties window and, under Data properties, enter the attribute MACHINE_ID in “Link Child Fields” and “Link Master Fields”.

At this point the form should look like that shown in the exercise and it should be functional.

d) There are several steps in the creation of these forms. The following steps are one way to proceed.

1. Create the PM_WORK_ORDER Form.

   This form is based on a single table: PM_WORK_ORDER. After creating the form, in order that the WO_DATE will automatically default to the current date, go to the table PM_WORK_ORDER and choose design view. Select WO_DATE field. The format selection should be “Short Date”. The default value is: =Date(). This will default the textbox value to the current date in short date format.
2. Add the “Close” command button to the form. The On Click event procedure should be as follows:

Private Sub Close_Button_Click()
    On Error GoTo Err_Close_Button_Click
    Dim stDocName As String
    stDocName = "MacroClose"
    DoCmd.RunMacro stDocName
    Exit_Close_Button_Click:
    Exit Sub
  Err_Close_Button_Click:
    MsgBox Err.Description
    Resume Exit_Close_Button_Click
End Sub

The Macro is created as follows:

<table>
<thead>
<tr>
<th>Action</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Close and save form</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Type</td>
</tr>
<tr>
<td>Object Name</td>
</tr>
<tr>
<td>Save</td>
</tr>
</tbody>
</table>

3. In order to open the PM_WORK_ORDER_FORM, move to a new record, populate the form with data from MONITOR_HOURS_OF_OPERATION form, and then close
the MONITOR_HOURS_OF_OPERATION form, the following code can be used for
the On Click event procedure of the “Make WO” command button.

Private Sub Make_WO_Button_Click()

Dim stDocName As String

stDocName = "MacroOpenWorkOrder"
DoCmd.RunMacro stDocName

Forms!(PM_WORK_ORDER)![MACHINE_ID].Value = Me.MACHINE_ID.Value
Forms!(PM_WORK_ORDER)![PM_TASK_ID].Value = Me.Text13.Value
Forms!(PM_WORK_ORDER)![MACHINE_HOURS].Value =
          Me.MACHINE_TOTAL_HOURS.Value

Forms!(MONITOR_HOURS_OF_OPERATION).SetFocus
stDocName = "MacroClose"
DoCmd.RunMacro stDocName

End Sub

The Macro named MacroOpenWorkOrder is created as follows:
<table>
<thead>
<tr>
<th>Action</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenForm</td>
<td>Open PM_WORK_ORDER</td>
</tr>
<tr>
<td>GoToRecord</td>
<td>Move to a new record in PM_WORK_ORDER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Type</td>
</tr>
<tr>
<td>Object Name</td>
</tr>
<tr>
<td>Record</td>
</tr>
<tr>
<td>Offset</td>
</tr>
</tbody>
</table>
CHAPTER 7 ANSWERS

7.1 The boundaries are determined through the interview process. It is usually documented by specifying the activities included within the boundaries using an activity model, such as IDEF0. See section 7.2.3.

7.2 The user view is the data required by a particular user or user group. It is determined through an interview process with the potential users of the information system. It is usually documented in a data model such as IDEF1X. See sections 7.2.2 and 7.3.2.

7.3 The business rules describe how the enterprise operates. Some business rules will define the cardinality relationships among entities in the data model. See section 7.3.2.

7.4 The entity BATCH enables us to trace a finished product lot to a lot of ingredients. Using the entity MATERIAL_LOT and its child, PURCHASED_LOT, the VENDOR_LOT_NO and VENDOR_ID can be determined. This gets us back to the immediate source in the supply chain. The FDA can then go to the vendor and, if the vendor has complied with the FDA regulation, lot tracing can continue to his immediate supply sources. The regulation requires that each enterprise can track backward to the immediate source and forward to the immediate customers.

7.5

7.6 Answers on instructor database, Lot_Trace_CH7_Instructor.

7.7 The answer is on the instructor database, Lot_Trace_CH7_Instructor.
CHAPTER 8 ANSWERS

8.1 This is a creative exercise meant to get the student interested in developing his or her own web page. There will be many different examples.

8.2 The following HTML files suffice to answer this question.

```html
<HTML>
<HEAD><TITLE>UNIVERSITY FOOD INC.</TITLE>
<FRAMESET COLS="20%,80%">
<FRAME SRC="LIST82.html" FRAMEBORDER=1>
<FRAME SRC="UniversityFood.html" NAME="right" FRAMEBORDER=1>
</FRAMESET>
</HTML>

<HTML>
<HEAD>
<TITLE>List</TITLE>
</HEAD>
<BODY>
<br><br>
<UL>
  <LI>Home
  <BR>  <LI>About us
  <BR>  <LI>Products
  <BR>  <LI>Location
  <BR>  <LI><A HREF="UniversityFoodContact82.html" TARGET="right">Contact us</A>
</UL>
</BODY>
</HTML>
```
8.3

<HTML>

<HEAD>
<TITLE>USF RENT-A-CAR</TITLE>
</HEAD>

<BODY>

<H1 ALIGN="CENTER"><I>USF RENT-A-CAR</I></H1 ALIGN="CENTER">

<br />

<B>REGISTRATION REQUEST FORM: (All fields are required)</B>

<form action="http://localhost/USF/Reservation85.asp" method="GET">

<P STYLE="margin-left:3.63in;font:10pt arial">NAME:<input type="text" name="company"></P>

<P STYLE="margin-left:2.8in;font:10pt arial">STREET ADDRESS:<input type="text" name="street"></P>

<P STYLE="margin-left:3.73in;font:10pt arial">CITY:<input type="text" name="city"></P>

<P STYLE="margin-left:3.6in;font:10pt arial">STATE:<input type="text" name="state"></P>

<P STYLE="margin-left:3.83in;font:10pt arial">ZIP:<input type="text" name="zip"></P>

<P STYLE="margin-left:3.26in;font:10pt arial">CAR CLASS:<select name="class" size=1>
<option>Compact</option>
<option>Standard</option>
<option>Full Size</option>
</select></P>

<P STYLE="margin-left:2.77in;font:10pt arial">PICKUP LOCATION: <select name="Location" size=1>
<option>Tampa</option>
<option>St. Petersburg</option>
<option>Orlando</option>
</select></P>

<P STYLE="margin-left:3.0in;font:10pt arial">PICKUP MONTH:<select name="PUMONTH" size=1>
<option>JAN</option>
<option>FEB</option>
</select></P>

</form>

<br />

Click <a href="http://www.universityfood.html">Here</a> to return to home page.

</BODY>

</HTML>
<OPTION> MAR <OPTION> APR <OPTION> MAY <OPTION> JUN <OPTION> JUL <OPTION> AUG <OPTION> SEP <OPTION> OCT <OPTION> NOV <OPTION> DEC </SELECT>
DAY: <SELECT NAME="PUDAY" SIZE=1>
1 <OPTION> 2 <OPTION> 3 <OPTION> 4 <OPTION> 5 <OPTION> 6
7 <OPTION> 8 <OPTION> 9 <OPTION> 10 <OPTION> 11 <OPTION> 12 <OPTION> 13 <OPTION> 14 <OPTION> 15 <OPTION> 16 <OPTION> 17 <OPTION> 18
19 <OPTION> 20 <OPTION> 21 <OPTION> 22 <OPTION> 23 <OPTION> 24 <OPTION> 25 <OPTION> 26
27 <OPTION> 28 <OPTION> 29 <OPTION> 30
31 <OPTION> </SELECT> YEAR: <SELECT NAME="PUYEAR" SIZE=1>
2006 <OPTION> 2007 <OPTION> 2008 </SELECT>
HOUR: <SELECT NAME="PUHOUR" SIZE=1>
12AM <OPTION> 1AM <OPTION> 2AM <OPTION> 3AM <OPTION> 4AM <OPTION> 5AM <OPTION> 6AM <OPTION> 7AM <OPTION> 8AM <OPTION> 9AM <OPTION> 10AM <OPTION> 11AM <OPTION> 12PM <OPTION> 1PM <OPTION> 2PM <OPTION> 3PM <OPTION> 4PM <OPTION> 5PM <OPTION> 6PM <OPTION> 7PM <OPTION> 8PM <OPTION> 9PM <OPTION> 10PM <OPTION> 11PM </SELECT>
</P>
</FORM>

<p style="margin-left:2.85in;font:10pt arial">DROPOFF MONTH: <SELECT NAME="DOMONTH" SIZE=1>
JAN <OPTION> FEB <OPTION> MAR <OPTION> APR <OPTION> MAY <OPTION> JUN <OPTION> JUL <OPTION> AUG <OPTION> SEP <OPTION> OCT <OPTION> NOV <OPTION> DEC </SELECT>
DAY: <SELECT NAME="DODAY" SIZE=1>
1 <OPTION> 2 <OPTION> 3 <OPTION> 4 <OPTION> 5 <OPTION> 6
7 <OPTION> 8 <OPTION> 9 <OPTION> 10 <OPTION> 11 <OPTION> 12 <OPTION> 13 <OPTION> 14 <OPTION> 15 <OPTION> 16 <OPTION> 17 <OPTION> 18
19 <OPTION> 20 <OPTION> 21 <OPTION> 22 <OPTION> 23 <OPTION> 24 <OPTION> 25 <OPTION> 26
27 <OPTION> 28 <OPTION> 29 <OPTION> 30
31 <OPTION> </SELECT> YEAR: <SELECT NAME="DOYEAR" SIZE=1>
2006 <OPTION> 2007 <OPTION> 2008 </SELECT> HOUR: <SELECT NAME="DOHOUR" SIZE=1>
12AM <OPTION> 1AM <OPTION> 2AM <OPTION> 3AM <OPTION> 4AM <OPTION> 5AM <OPTION> 6AM <OPTION> 7AM <OPTION> 8AM <OPTION> 9AM <OPTION> 10AM <OPTION> 11AM <OPTION> 12PM <OPTION> 1PM <OPTION> 2PM <OPTION> 3PM <OPTION> 4PM <OPTION> 5PM <OPTION> 6PM <OPTION> 7PM <OPTION> 8PM <OPTION> 9PM <OPTION> 10PM <OPTION> 11PM </SELECT>
</P>

<P STYLE="margin-left:2.85in;font:10pt arial">DISCOUNTS: NONE <INPUT TYPE="radio" NAME="discount" VALUE="None" CHECKED>
Senior <INPUT TYPE="radio" NAME="discount" VALUE="senior">
Veteran <INPUT TYPE="radio" NAME="discount" VALUE="veteran">
USF Alumni <INPUT TYPE="radio" NAME="discount" VALUE="alumni"></P>

</FORM>

</BODY>
</HTML>
8.4 You must put an ACTION statement in the code of 8.3, above to direct the server to the file in the Localhost. The code below will handle the response.

```html
<HTML>
<HEAD>
<TITLE>USF REGISTRATION</TITLE>
</HEAD>

<BODY>

<H1 ALIGN="CENTER">USF RENT-A-CAR</H1>

<H2 ALIGN="CENTER">RESERVATION INFORMATION</H2>

<H4 ALIGN="LEFT">You have submitted the following registration request:

```csharp
+% Response.Write(" Customer Name: ";&Request.QueryString("customer")&"<BR>
Response.Write(" Address: ";&Request.QueryString("street")&","
Response.Write(" ";&Request.QueryString("city")&", 
Response.Write(" ";&Request.QueryString("state")&", 
Response.Write(" zip"&"<BR>
Response.Write(" Car Type: ";&Request.QueryString("class")&"<BR>
Response.Write(" Pickup Location: ";&Request.QueryString("location")&"<BR>
Response.Write(" Pickup Date: ";&Request.QueryString("PUMONTH")&" 
Response.Write(" ";&Request.QueryString("PUDAY")&" 
Response.Write(" ";&Request.QueryString("PUYEAR")&" 
Response.Write(" ";&Request.QueryString("PUHOUR")&"<BR>
Response.Write(" Drop Off Date: ";&Request.QueryString("DOMONTH")&" 
Response.Write(" ";&Request.QueryString("DODAY")&" 
Response.Write(" ";&Request.QueryString("DOYEAR")&" 
Response.Write(" ";&Request.QueryString("DOHOUR")&"<BR>
Response.Write(" Discount Type: ";&Request.QueryString("discount")
```%>

</BODY>
</HTML>

8.5 The following ASP code will insert the reservation data into the database and return a simple statement that the reservation has been made successfully.

```html
<HTML>
<HEAD>
<TITLE>USF New Reservation</TITLE>
</HEAD>

<BODY>

```csharp

```
```
<H1 ALIGN="CENTER"><I>USF RENT-A-CAR</I></H1 ALIGN="CENTER">

<P>

<% 'Connect to and open the database

Set ConnectionToDatabase=Server.CreateObject("ADODB.Connection")
ConnectionToDatabase.ConnectionTimeout=30
strCnn="Provider=Microsoft.jet.OLEDB.4.0;Data Source=C:\MyDB\USF.mdb;"
ConnectionToDatabase.Open strCnn

'Create RecordSet Object

Set RecordSet=Server.CreateObject("ADODB.RecordSet")

'Execute SQL Command

RecordSet.Open"SELECT * FROM RESERVATION", ConnectionToDatabase,1,2

RecordSet.AddNew
RecordSet("Customer")=Request.QueryString("customer")
RecordSet("Street")=Request.QueryString("street")
RecordSet("City")=Request.QueryString("city")
RecordSet("State")=Request.QueryString("state")
RecordSet("ZIP")=Request.QueryString("zip")
RecordSet("Class")=Request.QueryString("class")
RecordSet("Location")=Request.QueryString("location")
RecordSet("PUMONTH")=Request.QueryString("PUMONTH")
RecordSet("PUDAY")=Request.QueryString("PUDAY")
RecordSet("PUYEAR")=Request.QueryString("PUYEAR")
RecordSet("PUHOUR")=Request.QueryString("PUHOUR")
RecordSet("DOMONTH")=Request.QueryString("DOMONTH")
RecordSet("DODAY")=Request.QueryString("DODAY")
RecordSet("DOYEAR")=Request.QueryString("DOYEAR")
RecordSet("DOHOUR")=Request.QueryString("DOHOUR")
RecordSet("Discount")=Request.QueryString("discount")
RecordSet.Update

Response.Write("Your Reservation has been Successfully Entered")

'Close and disconnect from the database

ConnectionToDatabase.Close
Set ConnectionToDatabase=Nothing

%>

</BODY>
</HTML>
8.6 Retrieving and returning the reservation number is done by adding the following code before closing the database.

```
DIM reservation
RecordSet.MoveNext
reservation=RecordSet("Reservation_no")
Response.Write("Your Reservation Number is ",reservation)
```

8.7

a) A typical XML file is as follows:

The reservations made on 1/5/2006

```
<?xml version="1.0"?>
<!-- File Name: RESERVATION_1_5_06.xml -->
<RESERVATION>
  <RESERVATION_RECORD>
    <Reservation_no>10</Reservation_no>
    <Customer>Thomas Brown</Customer>
    <Street>65 Douglas Drive</Street>
    <City>Douglastown</City>
    <State>NJ</State>
    <ZIP>08854</ZIP>
    <Class>Full Size</Class>
    <Location>Tampa</Location>
    <PUMONTH>1</PUMONTH>
    <PUDAY>8</PUDAY>
    <PUYEAR>2006</PUYEAR>
    <PUTIME>1PM</PUTIME>
    <DOMONTH>1</DOMONTH>
    <DODAY>10</DODAY>
    <DOYEAR>2006</DOYEAR>
    <DOTIME>1PM</DOTIME>
    <Discount>Senior</Discount>
  </RESERVATION_RECORD>
</RESERVATION>
```
b) A typical Schema is as follows:

```xml
<?xml version="1.0"?>
<!--File Name: RESERVATION Schema.xsd -->
xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="RESERVATION">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="RESERVATION_RECORD" minOccurs="1" maxOccurs="100"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

c) By placing the RESERVATION database in C:\Corporate_Database and the XML files in C:\Branch_XML and following the instructions in XML Exercise 8.4, the files will be imported into tables of the database. A new table is created under the name RESERVATION_RECORD. It would remain for corporate office to move the data into its enterprise tables.
Exercise 1: The following are the HTML files for Figure 8CS.1.

University Food is a leading manufacturer and distributor of gourmet edibles for the college dining halls of America. Partaking of a University Food meal is a gastronomical experience you will not easily forget. You can obtain University Food products only by factory direct ordering over the internet. You must be a qualified customer and have your password available for ordering. Happy eating! 

Please sign in:

Company: <input type="text" name="company"><br>
Password: <input type="password" name="passwd"><br>
Submit <input type="submit" value="Ok"></form>

Not a registered user, click <a href="NewCustomer.html">here</a>
Exercise 2: The HTML file for the right hand frame of Figure 8CS.2 is as follows:

```html
<HTML>
  <HEAD>
    <TITLE>List</TITLE>
  </HEAD>
  <BODY>
    <BR><BR><BR>
    <UL>
      <LI>Home
      <BR>
      <LI>About us
      <BR>
      <LI>Products
      <BR>
      <LI>Location
      <BR>
      <LI>Contact us
    </UL>
  </BODY>
</HTML>
```

Exercise 3: The following ASP file is used to evaluate the submitted data and respond with the appropriate browser page. This ASP file is called from Figure 8CS.2 using the ACTION.

```html
<HTML>
  <HEAD>
    <TITLE>UNIVERSITY FOODS INCORPORATED</TITLE>
  </HEAD>
  <BODY>
    <BR>
    <BR>
    <H1 ALIGN="CENTER">UNIVERSITY FOOD INCORPORATED</H1>
    <BR>
    <B>NEW CUSTOMER REGISTRATION: (All fields are required)</B><BR>
    <FORM ACTION="http://localhost/food/newcustomerdataanalysis.asp" METHOD="GET">
      <P STYLE="margin-left:2.9in;font:10pt arial">COMPANY NAME:<INPUT TYPE="Text"
      NAME="company"></P>
      <P STYLE="margin-left:2.8in;font:10pt arial">STREET ADDRESS:<INPUT TYPE="Text"
      NAME="street"></P>
      <P STYLE="margin-left:3.73in;font:10pt arial">CITY:<INPUT TYPE="Text" NAME="city"></P>
      <P STYLE="margin-left:3.6in;font:10pt arial">STATE:<INPUT TYPE="Text" NAME="state"></P>
      <P STYLE="margin-left:3.83in;font:10pt arial">ZIP:<INPUT TYPE="Text" NAME="zip"></P>
      <P STYLE="margin-left:2.85in;font:10pt arial">ROUTING NUMBER:<INPUT TYPE="Text"
      NAME="routing"></P>
      <P STYLE="margin-left:2.8in;font:10pt arial">ACCOUNT NUMBER:<INPUT TYPE="Text"
      NAME="account"></P>
      <P STYLE="margin-left:2.7in;font:10pt arial">CREATE PASSWORD:<INPUT TYPE="password"
      NAME="passwd"></P>
      <P STYLE="margin-left:2.6in;font:10pt arial">CONFIRM PASSWORD:<INPUT TYPE="password"
      NAME="passwd2"></P>
      <P STYLE="margin-left:3.0in"><INPUT TYPE="submit" value="submit registration"></P>
    </FORM>
  </BODY>
</HTML>
```
statement. The screens for Figures 8CS.3 through 8CS.6 are imbedded in the responses below. This file is a general outline for handling the requirements, but does not include detailed validity checking for data other than company name and password.

File name: NewCustomerDataAnalysis.asp

<HTML>
<HEAD>
<TITLE>UNIVERSITY FOODS INCORPORATED</TITLE>
</HEAD>
<BODY>
<H1 ALIGN="CENTER"><I>UNIVERSITY FOODS INCORPORATED</I></H1 ALIGN="CENTER">
<P ALIGN="CENTER">
<IMG SRC="foodpic.jpg">
</P>
<P>
<% 
DIM Flag_1 
DIM Flag_2 
DIM Flag_3 
Flag_1=0 
Flag_2=0 
Flag_3=0 
'Check that all text boxes are filled.
If Request.QueryString("Company")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("Street")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("City")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("State")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("Zip")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("routing")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("account")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("passwd")="" Then 
  Flag_1=1 
ElseIf Request.QueryString("passwd2")="" Then 
  Flag_1=1 
Else 
  Flag_1=0 
End If


'Check to see that the password and its confirmation are the same.

If Request.QueryString("passwd")=Request.QueryString("passwd2") Then
   Flag_2=0
Else
   Flag_2=1
End If

'Respond based on the first two checks.

If Flag_1=1 and Flag_2=0 Then
   Response.Write("Your Registration was Not successful. You must complete all registration fields.")
   Response.Write("<BR>")
   Response.Write("<A HREF='NewCustomer.html'>Click Here to Return to Registration Page</A>"
ElseIf Flag_1=0 and Flag_2=1 Then
   Response.Write("Your Registration was Not successful. Your password confirmation failed.")
   Response.Write("<BR>")
   Response.Write("Try Again!"")
   Response.Write("<BR>")
   Response.Write("<A HREF='NewCustomer.html'>Click Here to Return to Registration Page</A>"
ElseIf Flag_1=1 and Flag_2=1 Then
   Response.Write("Your Registration was Not successful. Your password confirmation failed and you did not complete all registration fields.")
   Response.Write("<BR>")
   Response.Write("Try Again!"")
   Response.Write("<BR>")
   Response.Write("<A HREF='NewCustomer.html'>Click Here to Return to Registration Page</A>"
ElseIf Flag_1=0 and Flag_2=0 Then
   'Check to see that the customer name and password are not already in use.

   'DEFINE CONNECTION INSTANCE
   Set ConnectionToDatabase=Server.CreateObject("ADODB.Connection")

   'ESTABLISH AND OPEN CONNECTION
   ConnectionToDatabase.ConnectionTimeout=30
   strcnn="Provider=Microsoft.jet.OLEDB.4.0;Data Source=C:\MyDB\ORDERS.mdb;"
   ConnectionToDatabase.Open strcnn

   'CREATE RECORDSET INSTANCE
   Set MyRecordSet=Server.CreateObject("ADODB.RecordSet")

   'OPEN RECORDSET AND QUERY DATABASE
   MyRecordSet.Open"SELECT * FROM CUSTOMER", ConnectionToDatabase,1,2

   'PERFORM DESIRED OPERATIONS ON RECORDSET
Do while NOT MyRecordSet.EOF
    If MyRecordSet("CUST_NAME")=Request.QueryString("company") OR 
        MyRecordSet("PASSWORD")=Request.QueryString("passwd") Then 
        Response.Write("Your Registration was not successful! ")
        Response.Write("The Customer Name and/or Password is already in use. Try again!")
        Response.Write("<BR>")
        Response.Write("<A HREF='NewCustomer.html'>Click Here to Return to Registration Page</A>"
        flag_3=1
    End If
    MyRecordSet.MoveNext
Loop

'Register the new customer.

If flag_3=0 Then 
    MyRecordSet.AddNew 
    MyRecordSet("CUST_NAME")=Request.QueryString("company")
    MyRecordSet("PASSWORD")=Request.QueryString("passwd")
    MyRecordSet("CUST_STREET")=Request.QueryString("street")
    MyRecordSet("CUST_CITY")=Request.QueryString("city")
    MyRecordSet("CUST_STATE")=Request.QueryString("state")
    MyRecordSet("CUST_ZIP")=Request.QueryString("zip")
    MyRecordSet("ROUT_NO")=Request.QueryString("routing")
    MyRecordSet("ACCT_NO")=Request.QueryString("account")
    MyRecordSet.Update
    Response.Write("Welcome ",Request.QueryString("company"),"!")
    Response.Write("<BR>")
    Response.Write("Your Registration was successful. ")
    Response.Write("<A HREF='UniversityFood.html'>Click Here to continue</A>"
End If

'CLOSE THE RECORDSET

MyRecordSet.Close
Set MyRecordSet=Nothing

'CLOSE CONNECTION TO DATABASE

ConnectionToDatabase.Close
Set ConnectionToDatabase=Nothing

End If

</BODY>
</HTML>
Exercise 4: The following ASP file provides the right hand frame of Figure 8CS.7 and other responses to the login from Figure 8CS.1.

**File name: LoginResponse.asp**

```html
<HTML>
<HEAD>
<TITLE>Order Confirmation</TITLE>
</HEAD>
<BODY>

<H1 ALIGN="CENTER"><I>UNIVERSITY FOOD INCORPORATED</I></H1 ALIGN="CENTER">

<P ALIGN="CENTER">

<IMG SRC="foodpic.jpg">

</P>

<P>

<%'

'Connect to and open the database

Set ConnectionToDatabase=Server.CreateObject("ADODB.Connection")
ConnectionToDatabase.ConnectionTimeout=30
strcnn="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\MyDB\ORDERS.mdb;"
ConnectionToDatabase.Open strcnn

'Create RecordSet Object

Set RecordSet=Server.CreateObject("ADODB.RecordSet")

'Execute SQL Command

RecordSet.Open"SELECT * FROM CUSTOMER", ConnectionToDatabase

DIM Flag
Flag=0

Do While NOT RecordSet.EOF
    If RecordSet("CUST_NAME")=Request.QueryString("company") And
    RecordSet("PASSWORD")=Request.QueryString("passwd") Then
        Response.Write("You are now logged in")
        Response.write("<BR>")
        Response.Write("<A HREF='UniversityFoodOrderSelect.html'>Click Here to continue</A>")
        Session("company")=Request.QueryString("company")
        Session("password")=Request.QueryString("passwd")
        Flag=1
    Else
        End If
    RecordSet.MoveNext
Loop

If Flag=0 Then
```

60
Response.Write("Your Login Failed! Check your Company Name and Password")
Response.Write("<BR>")
Response.Write("<A HREF='UniversityFood.html'>Click Here to return to login page</A>"
End If

' close Recordset

RecordSet.close
Set RecordSet=Nothing

' Close and disconnect from the database

ConnectionToDatabase.Close
Set ConnectionToDatabase=Nothing

%/>
</BODY>
</HTML>

The right hand frame of Figure 8CS.8 is the following HTML file.

Exercise 5: The HTML file for the order form is shown below as well as the ASP file for handling the order once it is placed.
File name OrderResponse.asp

```
<HTML>
<HEAD>
<TITLE>Order Confirmation</TITLE>
</HEAD>
<BODY>

<H1>University Food Products</H1>

<P>
'Connect to and open the database

Set ConnectionToDatabase=Server.CreateObject("ADODB.Connection")
ConnectionToDatabase.ConnectionTimeout=30
strcnn="Provider=Microsoft.jet.OLEDB.4.0;Data Source=C:\MyDB\ORDERS.mdb;"
ConnectionToDatabase.Open strcnn

'FIND THE CUSTOMER ID CORRESPONDING TO THE CUSTOMER PASSWORD

</P>
</BODY>
</HTML>
```
'Define RecordSet instance
Set MyRecordSet1=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database
    MyRecordSet1.Open"SELECT * FROM CUSTOMER", ConnectionToDatabase

DIM customer_ID
Do While NOT MyRecordSet1.EOF
    If MyRecordSet1("PASSWORD")=Session("password") Then
        customer_ID=MyRecordSet1("CUSTOMER_ID")
    End If
    MyRecordSet1.MoveNext
Loop

'Close RecordSet
    MyRecordSet1.Close
    Set MyRecordSet1=Nothing

'FIND THE LAST CUSTOMER ORDER NUMBER IN THE CUSTOMER_ORDER TABLE

'Define RecordSet instance
Set MyRecordSet2=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database
    MyRecordSet2.Open"SELECT * FROM CUSTOMER_ORDER", ConnectionToDatabase,1,2

'Perform Operations on the RecordSet

'RETRIEVE THE LAST ORDER NUMBER AND INCREMENT BY 1

DIM order_no

    MyRecordSet2.MoveLast
    order_no=MyRecordSet2("ORDER_NO")
    order_no=order_no+1

'INSERT THE NEW PO NUMBER INTO THE DATABASE

    MyRecordSet2.AddNew
    MyRecordSet2("ORDER_NO")=order_no
    MyRecordSet2("CUSTOMER_ID")=customer_ID
    MyRecordSet2("CUST_PO_NO")=Request.QueryString("po_num")
    MyRecordSet2.Update

'Close RecordSet
    MyRecordSet2.Close
    Set MyRecordSet2=Nothing

'INSERT NEW ORDER DETAILS INTO DATABASE
'Define RecordSet instance
Set MyRecordSet3=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database
MyRecordSet3.Open"SELECT * FROM ORDER_DETAIL", ConnectionToDatabase,1,2

'Perform Operations on the RecordSet
DIM J
J=1

'INSERT THE ORDER DETAIL FOR BEEF STEW INTO THE DATABASE
If Request.QueryString("bs_qty")<>"" Then

'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE
Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database
MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

DIM price
Do While NOT MyRecordSet4.EOF
    If MyRecordSet4("MATERIAL_ID")="905" Then
        price=MyRecordSet4("STD_UNIT_COST")
    End If
    MyRecordSet4.MoveNext
Loop

'Close RecordSet
MyRecordSet4.Close
Set MyRecordSet4=Nothing

MyRecordSet3.AddNew
MyRecordSet3("ORDER_NO")=order_no
MyRecordSet3("ORDER_LINE_ITEM")=J
MyRecordSet3("MATERIAL_ID")="905"
MyRecordSet3("MATL_QTY")=Request.QueryString("bs_qty")
MyRecordSet3("UNITS")="Cases"
MyRecordSet3("PRICE")=price
MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("bs_date")
MyRecordSet3("STATUS")="OPEN"
MyRecordSet3("STATE")="ON ORDER"
MyRecordSet3.Update
J=J+1
End If
'INSERT THE ORDER DETAIL FOR CHICKEN IN CREAM SAUCE INTO THE DATABASE

If Request.QueryString("cs_qty")<>"" Then

'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE

Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database

MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

Do While NOT MyRecordSet4.EOF

  If MyRecordSet4("MATERIAL_ID")="30" Then
    price=MyRecordSet4("STD_UNIT_COST")
  End If

  MyRecordSet4.MoveNext

Loop

'Close RecordSet
MyRecordSet4.Close
Set MyRecordSet4=Nothing

MyRecordSet3.AddNew
MyRecordSet3("ORDER_NO")=order_no
MyRecordSet3("ORDER_LINE_ITEM")=J
MyRecordSet3("MATERIAL_ID")="30"
MyRecordSet3("MATL_QTY")=Request.QueryString("cs_qty")
MyRecordSet3("UNITS")="Cases"
MyRecordSet3("PRICE")=price
MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("cs_date")
MyRecordSet3("STATUS")="OPEN"
MyRecordSet3("STATE")="ON ORDER"
MyRecordSet3.Update
J=J+1

End If

'INSERT THE ORDER DETAIL FOR CHICKEN NOODLE SOUP INTO THE DATABASE

If Request.QueryString("cN_qty")<>"" Then

'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE

Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database

MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

Do While NOT MyRecordSet4.EOF

J=J+1

End If
If MyRecordSet4("MATERIAL_ID")="333" Then
    price=MyRecordSet4("STD_UNIT_COST")
End If
MyRecordSet4.MoveNext
Loop

'Close RecordSet
MyRecordSet4.Close
Set MyRecordSet4=Nothing

MyRecordSet3.AddNew
MyRecordSet3("ORDER_NO")=order_no
MyRecordSet3("ORDER_LINE_ITEM")=J
MyRecordSet3("MATERIAL_ID")="333"
MyRecordSet3("MATL_QTY")=Request.QueryString("cn_qty")
MyRecordSet3("UNITS")="Cases"
MyRecordSet3("PRICE")=price
MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("cn_date")
MyRecordSet3("STATUS")="OPEN"
MyRecordSet3("STATE")="ON ORDER"
MyRecordSet3.Update
J=J+1
End If

'INSERT THE ORDER DETAIL FOR CHILI WITH BEANS INTO THE DATABASE
If Request.QueryString("cb_qty")<>"" Then

'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE
Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database
MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

Do While NOT MyRecordSet4.EOF
    If MyRecordSet4("MATERIAL_ID")="233" Then
        price=MyRecordSet4("STD_UNIT_COST")
    End If
    MyRecordSet4.MoveNext
Loop

'Close RecordSet
MyRecordSet4.Close
Set MyRecordSet4=Nothing

MyRecordSet3.AddNew
MyRecordSet3("ORDER_NO")=order_no
MyRecordSet3("ORDER_LINE_ITEM")=J
MyRecordSet3("MATERIAL_ID")="233"
MyRecordSet3("MATL_QTY")=Request.QueryString("cb_qty")
MyRecordSet3("UNITS")="Cases"
MyRecordSet3("PRICE")=price
MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("cb_date")
MyRecordSet3("STATUS")="OPEN"
MyRecordSet3("STATE")="ON ORDER"
MyRecordSet3.Update
J=J+1
End If

'INSERT THE ORDER DETAIL FOR CORN BEEF HASH INTO THE DATABASE

If Request.QueryString("ch_qty")<>""

'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE

Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database

MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

Do While NOT MyRecordSet4.EOF

If MyRecordSet4("MATERIAL_ID")="440" Then
    price=MyRecordSet4("STD_UNIT_COST")
End If
MyRecordSet4.MoveNext
Loop

'Close RecordSet

MyRecordSet4.Close
Set MyRecordSet4=Nothing

MyRecordSet3.AddNew
MyRecordSet3("ORDER_NO")=order_no
MyRecordSet3("ORDER_LINE_ITEM")=J
MyRecordSet3("MATERIAL_ID")="440"
MyRecordSet3("MATL_QTY")=Request.QueryString("ch_qty")
MyRecordSet3("UNITS")="Cases"
MyRecordSet3("PRICE")=price
MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("ch_date")
MyRecordSet3("STATUS")="OPEN"
MyRecordSet3("STATE")="ON ORDER"
MyRecordSet3.Update
J=J+1
End If

'INSERT THE ORDER DETAIL FOR MACARONI AND CHEESE INTO THE DATABASE

If Request.QueryString("mc_qty")<>""

'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE

Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")
'Open RecordSet and Query Database

    MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

Do While NOT MyRecordSet4.EOF

    If MyRecordSet4("MATERIAL_ID")="243" Then
        price=MyRecordSet4("STD_UNIT_COST")
    End If
    MyRecordSet4.MoveNext
Loop

'Close RecordSet

    MyRecordSet4.Close
    Set MyRecordSet4=Nothing

    MyRecordSet3.AddNew
    MyRecordSet3("ORDER_NO")=order_no
    MyRecordSet3("ORDER_LINE_ITEM")=J
    MyRecordSet3("MATERIAL_ID")="243"
    MyRecordSet3("MATL_QTY")=Request.QueryString("mc_qty")
    MyRecordSet3("UNITS")="Cases"
    MyRecordSet3("PRICE")=price
    MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("mc_date")
    MyRecordSet3("STATUS")="OPEN"
    MyRecordSet3("STATE")="ON ORDER"
    MyRecordSet3.Update
    J=J+1
End If

'INSERT THE ORDER DETAIL FOR MEAT LOAF INTO THE DATABASE

If Request.QueryString("ml_qty")<>"" Then

    'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE

    Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")

    'Open RecordSet and Query Database

        MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

    Do While NOT MyRecordSet4.EOF

        If MyRecordSet4("MATERIAL_ID")="25" Then
            price=MyRecordSet4("STD_UNIT_COST")
        End If
        MyRecordSet4.MoveNext
    Loop

    'Close RecordSet
    MyRecordSet4.Close
Set MyRecordSet4=Nothing

MyRecordSet3.AddNew
MyRecordSet3("ORDER_NO")=order_no
MyRecordSet3("ORDER_LINE_ITEM")=J
MyRecordSet3("MATERIAL_ID")="25"
MyRecordSet3("MATL_QTY")=Request.QueryString("ml_qty")
MyRecordSet3("UNITS")="Cases"
MyRecordSet3("PRICE")=price
MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("ml_date")
MyRecordSet3("STATUS")="OPEN"
MyRecordSet3("STATE")="ON ORDER"
MyRecordSet3.Update
J=J+1
End If

'INSERT THE ORDER DETAIL FOR SALSBURY STEAK INTO THE DATABASE
If Request.QueryString("ss_qty")<>"" Then

'GET THE PRICE OF THE MATERIAL FROM THE MATERIAL TABLE
Set MyRecordSet4=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database
MyRecordSet4.Open"SELECT * FROM MATERIAL", ConnectionToDatabase

Do While NOT MyRecordSet4.EOF

If MyRecordSet4("MATERIAL_ID")="20" Then
    price=MyRecordSet4("STD_UNIT_COST")
End If
MyRecordSet4.MoveNext
Loop

'Close RecordSet
MyRecordSet4.Close
Set MyRecordSet4=Nothing

MyRecordSet3.AddNew
MyRecordSet3("ORDER_NO")=order_no
MyRecordSet3("ORDER_LINE_ITEM")=J
MyRecordSet3("MATERIAL_ID")="20"
MyRecordSet3("MATL_QTY")=Request.QueryString("ss_qty")
MyRecordSet3("UNITS")="Cases"
MyRecordSet3("PRICE")=price
MyRecordSet3("PROMISED_DEL_DATE")=Request.QueryString("ss_date")
MyRecordSet3("STATUS")="OPEN"
MyRecordSet3("STATE")="ON ORDER"
MyRecordSet3.Update
J=J+1
End If
Exercise 6: The ASP file that returns the existing orders is shown below.

File name: ReviewExistingOrders.asp

```html
<HTML>
<HEAD>
<TITLE>Order Review</TITLE>
</HEAD>
<BODY>

<H1 ALIGN="CENTER">University Food Incorporated</H1 ALIGN="CENTER">

<H2 ALIGN="CENTER">Customer Order Review Page</H2 ALIGN="CENTER">

<%'

DIM customer_ID

Response.Write("The following are your Purchase Order numbers")
Response.write("<P>")

'Connect to and open the database

Set ConnectionToDatabase=Server.CreateObject("ADODB.Connection")
ConnectionToDatabase.ConnectionTimeout=30
strcnn="Provider=Microsoft.jet.OLEDB.4.0;Data Source=C:\MyDB\ORDERS.mdb;"
ConnectionToDatabase.Open strcnn

'FIND THE CUSTOMER ID CORRESPONDING TO THE CUSTOMER PASSWORD

'Define RecordSet instance

Set MyRecordSet1=Server.CreateObject("ADODB.RecordSet")
```
'Open RecordSet and Query Database
    MyRecordSet1.Open"SELECT * FROM CUSTOMER", ConnectionToDatabase

Do While NOT MyRecordSet1.EOF
    If MyRecordSet1("PASSWORD")=Session("password") Then
        customer_ID=MyRecordSet1("CUSTOMER_ID")
    End If
    MyRecordSet1.MoveNext
Loop

'Close RecordSet
    MyRecordSet1.Close
    Set MyRecordSet1=Nothing

'FIND THE CUSTOMER ORDER NUMBERS CORRESPONDING TO THE CUSTOMER ID

'Define RecordSet instance
    Set MyRecordSet2=Server.CreateObject("ADODB.RecordSet")

'Open RecordSet and Query Database
    MyRecordSet2.Open"SELECT * FROM CUSTOMER_ORDER", ConnectionToDatabase

'Perform Operations on the RecordSet
Do While NOT MyRecordSet2.EOF
    If MyRecordSet2("CUSTOMER_ID")=customer_ID Then
        Response.Write(MyRecordSet2("CUST_PO_NO"))
        Response.Write("<BR>")
    End If
    MyRecordSet2.MoveNext
Loop

'Close RecordSet
    MyRecordSet2.Close
    Set MyRecordSet2=Nothing

'Close and disconnect from the database
    ConnectionToDatabase.Close
    Set ConnectionToDatabase=Nothing
%>

<P>
Return to homepage: <A HREF="UniversityFood.html">University Foods Home</A>
</P>
</BODY>
</HTML>
CHAPTER 9 ANSWERS

9.1

Normal Scenario:

1. The customer requests a performance and seating preference from the ticket sale system.
2. The system verifies whether or not the performance and seating are available.
3. The performance and seating are available. Inform customer.

Alternative Scenario:

1. The customer requests a performance and seating preference from the ticket sale system.
2. The system verifies whether or not the performance and seating are available.
3. The performance or seating are not available. Inform customer.

9.2 There are several possible additional scenarios. Here we illustrate a few additional cases besides the two in the text.
Scenario 3:

1. Customer requests a performance and seating from ticket sale system.
2. The system verifies whether the performance and seating are available.
3. The performance and seating are available. System offers the ticket to the customer.
4. The customer declines the ticket.
5. System informs the customer it is terminating the transaction.

Scenario 4:

1. Customer requests a performance and seating from ticket sale system.
2. The system verifies whether the performance and seating are available.
3. The performance and seating are available. System offers the ticket to the customer.
4. The customer accepts the ticket.
5. System temporarily places the ticket on hold.
6. The system requests credit card information from the customer.
7. Credit card information supplied by customer.
8. Credit card information not OK (fields not filled in or card number format wrong).
   System makes second request.
9. Credit card information supplied by customer.
10. Credit card information not OK. System makes third request.
11. Credit card information supplied by customer.
12. Credit card information not OK.
13. System informs customer it is terminating the transaction.

Scenario 5:

1. Customer requests a performance and seating from ticket sale system.
2. The system verifies whether the performance and seating are available.
3. The performance and seating are available. System offers the ticket to the customer.
4. The customer accepts the ticket.
5. System temporarily places the ticket on hold.
6. The system requests credit card information from the customer.
7. Credit card information supplied by customer. Information OK.
8. System requests authorization to debit the credit card from the authorization service.
9. The authorization service denies the request.
10. System informs the user of the failure and terminates the transaction.

9.3

1. **Normal Scenario:**

1. Customer requests cancellation (Transaction_ID, Name, Address).
2. System verifies transaction < 24 hours old.
3. Transaction < 24 hours = True. System asks customer to confirm cancellation request.
4. Customer confirms cancellation request.
5. System requests a credit to the Mastercard account from the authorization service.
6. Authorization service approves the credit.
7. System cancels sale.
8. System releases ticket(s).
9. System confirms the sale is cancelled to the customer.

b)
**e)**

![Diagram of Create Sale process]

- Create Sale
- Do Create_Sale
- Active Sale
- Cancel Sale
- Sale Active

**f)**

<table>
<thead>
<tr>
<th>Sale</th>
<th>Ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustName:string</td>
<td>Performance:string</td>
</tr>
<tr>
<td>CustAddress:string</td>
<td>Location:string</td>
</tr>
<tr>
<td>CreditCardType:string</td>
<td>Price:currency</td>
</tr>
<tr>
<td>CreditCardNumber:string</td>
<td>Status:string</td>
</tr>
<tr>
<td>CreateSale ()</td>
<td>VerifyAvailable ()</td>
</tr>
<tr>
<td>CancelSale ()</td>
<td>PlaceOnHold ()</td>
</tr>
</tbody>
</table>

**9.4**

![Diagram of Create PO process]

- Create PO
- Do Create_PO
- OPEN
- Close PO
- CLOSED
9.5
a)

Success Scenario:

1. Customer opens session (URL).
2. Server downloads web page (UserID3.html)
3. Customer enters data (company, passwd).
4. Data is submitted (company, passwd).
5. Server calls ASP file (Retrieval3.asp).
6. Middleware (IIS) creates database connection (ORDERS.mdb).
7. Connection confirmed.
8. Middleware processes query and opens recordset (MyRecordset).
9. Recordset retrieved.
10. ASP file executes procedure on recordset, obtains result (Flag = 0).
11. Server responds to Browser (company, passwd).
12. Database connection closed.
13. Customer closes session.
Failure Scenario:

1. Customer opens session (URL).
3. Customer enters data (company, passwd).
4. Data is submitted (company, passwd).
5. Server calls ASP file (Retrieval3.asp).
6. Middleware (IIS) creates database connection (ORDERS.mdb).
7. Connection confirmed.
8. Middleware processes query and opens recordset (MyRecordset).
9. Recordset retrieved.
10. ASP file executes procedure on recordset, obtains result (Flag = 1).
11. Server responds to Browser (Failure).
12. Database connection closed.
13. Customer closes session.
b)

9.6

a)

Note: In this description the Service Representative is directly interfacing with the system (database). The customer interacts with the Service Representative and, hence, is not shown. This also assumes that prior reservations are made by calling a Reservation Agent. Again, the customer does not interact with the system.
b) **Scenario 1, Prior Reservation:** From the description in Exercise 3.4.

1. Check for reservation (customer name).
2. Database responds reservation = True.
3. Service Representative inserts contract information (Lic. No., etc.)
4. Service Representative queries for available cars (Class).
5. Database returns available cars.
6. Agent assigns car (VIN) to contract.
7. Agent prints rental agreement.

**Scenario 2, No Prior Reservation:** From the description in Exercise 3.4.

1. Check for reservation (customer name).
2. Database responds reservation = False.
3. Service Representative inserts reservation information (customer name, etc.)
4. Service Representative inserts contract information (Lic. No., etc.)
5. Service Representative queries for available cars (Class).
6. Database returns available cars.
7. Agent assigns car (VIN) to contract.
8. Agent prints rental agreement.
c) In this response some general knowledge and intuition helps. When a car is first put into service by the company, it is created in the database. At that time it is available to be rented. It will circulate among three states. When it is in service, it will be either Available or Rented. The other state is when it is withdrawn from service for Maintenance.
CHAPTER 10 ANSWERS

10.1

<table>
<thead>
<tr>
<th>Current State</th>
<th>True Conditions</th>
<th>Events</th>
<th>Action</th>
<th>Resultant State</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) B</td>
<td>-</td>
<td>m</td>
<td>-</td>
<td>C</td>
</tr>
<tr>
<td>b) B</td>
<td>-</td>
<td>n</td>
<td>-</td>
<td>D</td>
</tr>
<tr>
<td>c) C</td>
<td>p</td>
<td>k</td>
<td>-</td>
<td>C</td>
</tr>
<tr>
<td>d) D</td>
<td>p</td>
<td>k</td>
<td>X</td>
<td>B</td>
</tr>
<tr>
<td>e) C</td>
<td>q</td>
<td>k</td>
<td>-</td>
<td>E</td>
</tr>
</tbody>
</table>

10.2

<table>
<thead>
<tr>
<th>Current State</th>
<th>True Conditions</th>
<th>Events</th>
<th>Action</th>
<th>Resultant State</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) D, H</td>
<td>-</td>
<td>k</td>
<td>-</td>
<td>E, F</td>
</tr>
<tr>
<td>b) E, F</td>
<td>p</td>
<td>m</td>
<td>X</td>
<td>E, G</td>
</tr>
<tr>
<td>c) E, G</td>
<td>-</td>
<td>n</td>
<td>-</td>
<td>E, H</td>
</tr>
<tr>
<td>d) E, G</td>
<td>p</td>
<td>n</td>
<td>-</td>
<td>D, G</td>
</tr>
<tr>
<td>e) E, H</td>
<td>p</td>
<td>n</td>
<td>-</td>
<td>D, H</td>
</tr>
</tbody>
</table>

10.3

a) This is a Begin on Commit dependency which is a Strong Causal type dependency

b) This is a Forced Commit on Abort dependency which is a Weak Causal type dependency

c) This is a Weak Begin on Commit dependency which is a Precedence type dependency
d) This is an External dependency which requires executing the task 3 times before it commits at which point it aborts.

e) This is a Value dependency where a parameter T must be greater than 5 before the task can be committed.
f) This is a combination of Begin on Abort and Forced Commit on Abort dependencies which are Strong and Weak Causal dependencies respectively.

10.4 The added Abort dependency must require that Commit sale aborts if process payment aborts.
Process Payment AND Commit Sale

Process Payment

Uncontrollable Transition

- Executing
- Precommit

Aborted
Commit

Commit Sale

Uncontrollable Transition

- Executing
- Precommit

Abort

Commit

(Process Payment has terminated AND Process Payment has Not Committed)
(Process Payment has terminated AND Process Payment has not aborted)
**CASE STUDIES**

10.6 a) The additional set of tasks are Check Reservation, Confirm Cancellation and Credit Payment. The additional set of actions is Inform Customer of invalid transaction ID or improper request and Update Status of Seats. Note that this is only one of many possible solutions.

b)  

```
Check Reservation  Begin on Commit  Confirm Cancellation  Begin on Commit  Credit Payment
```

c)
10.7 a) The set of tasks are Process Payment, Schedule Production, Select Shipper, and Cancel Sale. The set of Actions include Inform Customer of Payment Failure, Inform Customer of Final Sale, Inform Customer of Cancelled Order. Note that this is only one of many possible solutions.

b)
c)

University Food On-Line Order Processing

On-line order received → Process Payment & Schedule Production → Select Shipper

[Schedule Production Commits] → [Select Shipper Commits]

[Process Payment Aborts] → [Select Shipper Aborts]

Cancel Sale →

[Select Shipper Aborts] → [Select Shipper Commits]

d-e)