Homework 6- SOLUTIONS

1. Writing Observations Explicitly

```
data work.extended;
   set orion.discount;
   drop unit_sales_price;
   where Start_Date='01dec2011'd;
   Promotion='Happy Holidays';
   Season='Winter';
   output;
   Start_Date='01jul2012'd;
   End_Date='31jul2012'd;
   Season='Summer';
   output;
run;

title 'All discount ranges with the Happy Holidays promotion';
proc print data=work.extended;
run;
title;
```

2. Creating Multiple SAS Data Sets with Derived Values

```
data work.fast work.slow work.veryslow;
   set orion.orders;
   where Order_Type in (2,3);
   /* There are several correct ways to write this WHERE statement */
   ShipDays=Delivery_Date-Order_Date;
   if ShipDays<3 then output work.fast;
   else if 5<=ShipDays<=7 then output work.slow;
   else if ShipDays>7 then output work.veryslow;
   drop Employee_ID;
run;

title 'Orders taking more than 7 days to deliver';
proc print data=work.veryslow;
run;
title;
```

3. Specifying Variables and Observations

```
data work.instore (keep=Order_ID Customer_ID Order_Date)
     work.delivery (keep=Order_ID Customer_ID Order_Date ShipDays);
   set orion.orders (obs=30);
   where Order_Type=1;
   ShipDays=Delivery_Date-Order_Date;
   if ShipDays=0 then output work.instore;
   else if ShipDays>0 then output work.delivery;
run;
```

```
data work.instore (keep=Order_ID Customer_ID Order_Date)
     work.delivery (keep=Order_ID Customer_ID Order_Date ShipDays);
   set orion.orders;
   where Order_Type=1;
   ShipDays=Delivery_Date-Order_Date;
   if ShipDays=0 then output work.instore;
   else if ShipDays>0 then output work.delivery;
run;

title 'Deliveries from In-store Purchases';
proc print data=work.delivery;
run;
title;

title 'In-stock Store Purchases, By Year';
proc freq data=work.instore;
   tables Order_Date;
   format Order_Date year.;
run;
title;
```

Part II

1. Creating Accumulating Totals with Conditional Logic

```
data work.typetotals;
   set orion.order_fact (obs=10);
   where year(Order_Date)=2009;
   /* There are equivalent WHERE statements that would work */
   if Order_Type=1 then TotalRetail+Quantity;
   else if Order_Type=2 then TotalCatalog+Quantity;
   else if Order_Type=3 then TotalInternet+Quantity;
run;

proc print data=work.typetotals;
run;

data work.typetotals;
   set orion.order_fact;
   where year(Order_Date)=2009;
   /* There are equivalent WHERE statements that would work */
   if Order_Type=1 then TotalRetail+Quantity;
   else if Order_Type=2 then TotalCatalog+Quantity;
   else if Order_Type=3 then TotalInternet+Quantity;
   keep Order_ID Order_Date TotalRetail
        TotalCatalog TotalInternet;
```

```
run;

title '2009 Accumulating Totals for Each Type of Order';
proc print data=work.typetotals;
run;
title;
```

2. Summarizing and Grouping Data Using the DATA Step

```
proc sort data=orion.order_qtrsum out=work.custsort;
   by Customer_ID Order_Qtr;
run;

data work.qtrcustomers;
   set work.custsort;
   by Customer_ID Order_Qtr;
   if first.Order_Qtr=1 then do;
      Total_Sales=0;
      Num_Months=0;
   end;
   Total_Sales+Sale_Amt;
   Num_Months+1;
   if last.Order_Qtr=1;
   keep Customer_ID Order_Qtr Total_Sales Num_Months;
run;

title 'Total Sales to each Customer for each Quarter';
proc print data=work.qtrcustomers;
   format Total_Sales dollar11.2;
run;
title;
```

Part III

1. Using Conditional Logic to Output Multiple Observations

```
data work.lookup;
   set orion.country;
   Outdated='N';
   output;
   if Country_FormerName ne ' ' then do;
      Country_Name=Country_FormerName;
```

```
         Outdated='Y';
         output;
   end;
   drop Country_FormerName Population;
run;

title 'Current and Outdated Country Name Data';
proc print data=work.lookup;
run;
title;
```

3. **Identifying Extreme Values in Each Group of Data**

```
proc sort data=orion.customer_dim out=work.customers;
   by Customer_Type;
run;

data work.agecheck;
   set work.customers;
   by Customer_Type;
   retain oldest youngest o_ID y_ID;
   if first.Customer_Type=1 then do;
      oldest=Customer_BirthDate;
      youngest=Customer_BirthDate;
      o_ID=Customer_ID;
      y_ID=Customer_ID;
   end;
   if Customer_BirthDate < oldest then do;
      o_ID=Customer_ID;
      oldest=Customer_BirthDate;
   end;
   else if Customer_BirthDate > youngest then do;
      y_ID=Customer_ID;
      youngest=Customer_BirthDate;
   end;
   if last.Customer_Type=1 then do;
      agerange=(youngest-oldest)/365.25;
   output;
   end;
   keep Customer_Type oldest youngest o_ID y_ID agerange;
run;

title 'Oldest and Youngest Customers of each Customer Type';
proc print data=work.agecheck noobs;
   format oldest youngest date9. agerange 5.1;
run;
title;
```

Alternate Solution

```
proc sort data=orion.customer_dim out=work.customers;
   by Customer_Type Customer_BirthDate;
run;

data work.agecheck;
   set work.customers;
   by Customer_Type;
   /* Could instead use: by Customer_Type Customer_BirthDate;
      In this DATA step, either BY statement works. */
   retain oldest youngest o_ID y_ID;
   if first.Customer_Type=1 then do;
      o_ID=Customer_ID;
      oldest=Customer_BirthDate;
   end;
   /* Having sorted also on Customer_BirthDate, we know the first
   customer in each BY group will be the oldest (have the
   smallest SAS date value for a Birthday). */
   if last.Customer_Type=1 then do;
      y_ID=Customer_ID;
      youngest=Customer_BirthDate;
      agerange=(youngest-oldest)/365.25;
      output;
   end;
   /* Similar story: last in each BY group will be the youngest. */
   keep Customer_Type oldest youngest o_ID y_ID agerange;
run;

title 'Oldest and Youngest Customers of each Customer Type';
proc print data=work.agecheck noobs;
   format oldest youngest date9. agerange 5.1;
run;
title;
```