Erik Gonzalez-DeWhitt
CSCI-N311
Final Project
August 10, 2014

1. The purpose of the project is to create a back-end database for an online store selling Brazilian Jiu-Jitsu and mixed martial arts products. Along with collecting and holding standard information concerning product information, customer information, and also includes shipping information and considers valid promotional code entry.

   The project is meant for individuals working to manage the database and to easily add and remove entries easily. It's also meant to be able to view information related to the customer behavior and promotional code usage. Also, all sales must consider the agreed upon discount rates given to corresponding customers.

   Note: Customer_INFO names based on comic book characters and famed author Neil Gaiman.

2. Create Tables

a. Create

Worksheet | Query Builder

```
create table PRODUCT_INFO (
  PRODUCT_ID            NUMBER NOT NULL,
  DATE_ADDED            DATE,
  PRODUCT_NAME          VARCHAR2(20),
  PRICE                 NUMBER(5,2),
  COLOR                 VARCHAR2(8),
  FRAGILE               CHAR check (FRAGILE in ('Y', 'N')) ,
  constraint PRODUCT_PK PRIMARY KEY (PRODUCT_ID)
);
```

Script Output

Task completed in 2.312 seconds

```
table PRODUCT_INFO created.
```

Worksheet | Query Builder

```
create table CUSTOMER_INFO (
  CUSTOMER_ID            NUMBER NOT NULL,
  CUSTOMER_NAME          VARCHAR2(20),
  ADDRESS                VARCHAR2(20),
  CITY                   VARCHAR2(20),
  STATE                  VARCHAR2(2),
  PHONE                  VARCHAR2(16),
  EMAIL                  VARCHAR2(30),
  DISCOUNT               NUMBER(3,2),
  constraint CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID),
  constraint CUSTOMER_EMAIL_UQ UNIQUE (EMAIL)
);
```

Script Output

Task completed in 0.145 seconds

```
table CUSTOMER_INFO created.
```

Worksheet | Query Builder

```
create table SHIPPING_INFO (
  SHIPPING_TYPE          VARCHAR2(8),
  DAYS_TO_DELIVERY       NUMBER(2),
  constraint SHIPPING_UQ_TYPE UNIQUE (SHIPPING_TYPE)
);
```

Script Output

Task completed in 0.193 seconds

```
table SHIPPING_INFO created.
```

Worksheet | Query Builder

```
create table PROMO_CODE (
  PROMO_ID                NUMBER NOT NULL,
  DATE_START              DATE NOT NULL,
  DATE_END                DATE NOT NULL,
  CODE                    VARCHAR2(8) NOT NULL,
  LOCATIONS               VARCHAR2(25),
  constraint PROMO_PK PRIMARY KEY (PROMO_ID),
  constraint PROMO_UQ_CODE UNIQUE (CODE)
);
```

Script Output

Task completed in 4.523 seconds

```
00907. 00000 -  "missing right parenthesis"
*Cause:
*Action:
table PROMO CODE created.
```

Worksheet | Query Builder

```
create table ORDER_HISTORY (
  ORDER_ID                NUMBER NOT NULL,
  ORDER_DATE              DATE NOT NULL,
  CUSTOMER                NUMBER NOT NULL,
  PRODUCT                 NUMBER NOT NULL,
  COLOR                   VARCHAR2(8),
  QUANTITY                NUMBER(5),
  SUBTOTAL                NUMBER(8,2),
  TOTAL                   NUMBER(8,2),
  SHIPPING                VARCHAR2(8),
  PROMO                   VARCHAR2(8),
  constraint ORDER_PK PRIMARY KEY (ORDER_ID),
  constraint ORDER_FK_CUSTOMER FOREIGN KEY (CUSTOMER) references CUSTOMER_INFO(CUSTOMER_ID),
  constraint ORDER_FK_PRODUCT FOREIGN KEY (PRODUCT) references PRODUCT_INFO(PRODUCT_ID),
  constraint ORDER_FK_SHIPPING FOREIGN KEY (SHIPPING) references SHIPPING_INFO(SHIPPING_TYPE),
  constraint ORDER_FK_PROMO FOREIGN KEY (PROMO) references PROMO_CODE(CODE)
);
```

Script Output

Task completed in 0.144 seconds

```
table ORDER_HISTORY created.
```

Worksheet | Query Builder

```
create table SHIPPING_HISTORY (
  ORDER_ID                NUMBER NOT NULL,
  SHIP_DATE               DATE NOT NULL,
  SHIP_TYPE               VARCHAR2(8),
  ESTIMATED_ARRIVAL       DATE,
  constraint SHIPPING_FK_ORDER FOREIGN KEY (ORDER_ID) references ORDER_HISTORY(ORDER_ID),
  constraint SHIPPING_FK_SHIP FOREIGN KEY (SHIP_TYPE) references SHIPPING_INFO(SHIPPING_TYPE)
);
```

Script Output | Task completed in 1.117 seconds

```
table SHIPPING_HISTORY created.
```

b. Insert

Worksheet | Query Builder

```
insert all
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE, COLOR, FRAGILE)
    values (0, TO_DATE('01-JAN-01'), 'WONDER AIR GI', 108.95, 'WHITE', 'N')
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE, COLOR, FRAGILE)
    values (1, TO_DATE('01-JAN-01'), 'WONDER AIR GI', 118.95, 'BLUE', 'N')
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE, COLOR, FRAGILE)
    values (3, TO_DATE('01-JAN-01'), 'WONDER AIR GI', 128.95, 'BLACK', 'N')
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE, COLOR, FRAGILE)
    values (4, TO_DATE('03-FEB-01'), 'WONDER ELITE GI', 189.10, 'WHITE', 'N')
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE, COLOR, FRAGILE)
    values (5, TO_DATE('03-FEB-01'), 'WONDER ELITE GI', 199.95, 'BLACK', 'N')
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE)
    values (6, TO_DATE('01-MAR-02'), 'WONDER BOKKEN', 39.95)
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE)
    values (7, TO_DATE('02-MAR-02'), 'WONDER BO STAFF', 39.95)
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE, COLOR, FRAGILE)
    values (8, TO_DATE('01-JUL-13'), 'GALVAO SIGNED GI', 499.95, 'WHITE', 'Y')
  into PRODUCT_INFO (PRODUCT_ID, DATE_ADDED, PRODUCT_NAME, PRICE, COLOR, FRAGILE)
    values (9, TO_DATE('01-JUL-13'), 'GALVAO SIGNED GI', 499.95, 'BLACK', 'Y')
select * from DUAL;
```

Script Output | Query Result | Task completed in 2.767 seconds

```
9 rows inserted.
```

Worksheet | Query Builder

```
insert all
  into PROMO_CODE (PROMO_ID, DATE_START, DATE_END, CODE)
    values (0, TO_DATE('01-JAN-01'), TO_DATE('01-MAR-01'), 'WNDRYRS')
  into PROMO_CODE (PROMO_ID, DATE_START, DATE_END, CODE, LOCATIONS)
    values (1, TO_DATE('01-MAR-03'), TO_DATE('01-AUG-03'), 'WNDRBLL', 'NOT NY')
  into PROMO_CODE (PROMO_ID, DATE_START, DATE_END, CODE, LOCATIONS)
    values (2, TO_DATE('01-JUL-10'), TO_DATE('01-JUL-11'), 'WNDRFLL', 'CA')
select * from DUAL;
```

Script Output | Query Result | Task completed in 1.188 seconds

```
3 rows inserted.
```

Worksheet | Query Builder

```
insert all
  into SHIPPING_INFO (SHIPPING_TYPE, DAYS_TO_DELIVERY) values ('REG', 5)
  into SHIPPING_INFO (SHIPPING_TYPE, DAYS_TO_DELIVERY) values ('EXPR', 3)
  into SHIPPING_INFO (SHIPPING_TYPE, DAYS_TO_DELIVERY) values ('NIGHT', 1)
select * from DUAL;
```

Script Output

Task completed in 4.992 seconds

```
00928. 00000 -  "missing SELECT keyword"
*Cause:
*Action:
Error starting at line : 1 in command -
insert all
  into SHIPPING_INFO values ('REG', 5)
  into SHIPPING_INFO values ('EXPR', 3)
  into SHIPPING_INFO values ('NIGHT', 1)
Error at Command Line : 4 Column : 40
Error report -
SQL Error: ORA-00928: missing SELECT keyword
00928. 00000 -  "missing SELECT keyword"
*Cause:
*Action:
3 rows inserted.
```

Worksheet | Query Builder

```
insert all
  into Customer_INFO (Customer_ID, CUSTOMER_NAME, CITY, STATE, PHONE, EMAIL, DISCOUNT)
    values (0, 'BARRY ALLEN', 'STAR CITY', 'OK', '1-800-225-2277', 'BEST-FLASH@GMAIL.COM', 0.20)
  into Customer_INFO (Customer_ID, CUSTOMER_NAME, CITY, STATE, PHONE, EMAIL, DISCOUNT)
    values (1, 'WALLY WEST', 'STAR CITY', 'OK', '1-800-225-9255', 'NEXT-BEST-FLASH@GMAIL.COM', 0.15)
  into Customer_INFO (Customer_ID, CUSTOMER_NAME, ADDRESS, CITY, STATE, PHONE, EMAIL, DISCOUNT)
    values (2, 'DICK GRAYSON', 'ROOFTOPS', 'BLOODHAVEN', 'NY', '1-800-648-9464', 'NIGHTWING@GMAIL.COM', 0.50)
  into Customer_INFO (Customer_ID, CUSTOMER_NAME, ADDRESS, CITY, STATE, PHONE, EMAIL, DISCOUNT)
    values (3, 'JASON TODD', '123 DEAD WAY', 'GOTHAM', 'NY', '1-800-223-7627', 'BAD-ROBS@GMAIL.COM', 0.00)
  into Customer_INFO (Customer_ID, CUSTOMER_NAME, ADDRESS, PHONE, EMAIL, DISCOUNT)
    values (4, 'THE MONITOR', 'SPACE','1-800-666-4867', 'MONITOR@GMAIL.COM', 0.10)
  into Customer_INFO (Customer_ID, CUSTOMER_NAME, ADDRESS, DISCOUNT)
    values (5, 'DREAM', 'DREAMS', 1.00)
  into Customer_INFO (Customer_ID, CUSTOMER_NAME, ADDRESS, DISCOUNT)
    values (6, 'DEATH', 'AFTER LIFE', 1.00)
select * from DUAL;
```

Query Result | Script Output

Task completed in 0.08 seconds

```
7 rows inserted.
```

Worksheet | Query Builder

```
insert all
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (0, TO_DATE('03-JAN-01'), 0, 0, 'REG', 'WNDRYRS')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (1, TO_DATE('03-JAN-01'), 0, 3, 'REG', 'WNDRYRS')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (2, TO_DATE('03-MAR-03'), 2, 1, 'EXPR', 'WNDRBLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (3, TO_DATE('03-MAR-03'), 2, 5, 'EXPR', 'WNDRBLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (4, TO_DATE('03-JUL-03'), 5, 4, 'NIGHT', 'WNDRBLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (5, TO_DATE('03-JUL-03'), 5, 5, 'NIGHT', 'WNDRBLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (6, TO_DATE('03-JUL-03'), 6, 5, 'NIGHT', 'WNDRBLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (7, TO_DATE('23-AUG-10'), 1, 6, 'REG', 'WNDRFLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (8, TO_DATE('23-AUG-10'), 1, 7, 'REG', 'WNDRFLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING, PROMO)
    values (9, TO_DATE('24-AUG-10'), 1, 5, 'REG', 'WNDRFLL')
  into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SHIPPING)
    values (10, TO_DATE('24-AUG-13'), 5, 9, 'NIGHT')
select * from DUAL;
```

Script Output | Query Result

Task completed in 0.099 seconds

```
11 rows inserted.
```

Worksheet | Query Builder

```
insert all
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (0, TO_DATE('04-JAN-01'), 'REG', TO_DATE('09-JAN-01'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (1, TO_DATE('04-JAN-01'), 'REG', TO_DATE('09-JAN-01'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (2, TO_DATE('03-MAR-03'), 'EXPR', TO_DATE('05-MAR-03'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (3, TO_DATE('03-MAR-03'), 'EXPR', TO_DATE('05-MAR-03'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (4, TO_DATE('03-JUL-03'), 'NIGHT', TO_DATE('04-JUL-03'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (5, TO_DATE('03-JUL-03'), 'NIGHT', TO_DATE('04-JUL-03'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (6, TO_DATE('03-JUL-03'), 'NIGHT', TO_DATE('04-JUL-03'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (7, TO_DATE('23-AUG-10'), 'REG', TO_DATE('28-AUG-10'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (8, TO_DATE('23-AUG-10'), 'REG', TO_DATE('28-AUG-10'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (9, TO_DATE('24-AUG-10'), 'REG', TO_DATE('29-AUG-10'))
  into SHIPPING_HISTORY (ORDER_ID, SHIP_DATE, SHIP_TYPE, ESTIMATED_ARRIVAL)
    values (10, TO_DATE('24-AUG-13'), 'NIGHT', TO_DATE('25-AUG-13'))
select * from DUAL;
```

Script Output | Query Result

Task completed in 0.157 seconds

```
11 rows inserted.
```

3. Views

Worksheet | Query Builder

```
create or replace view CUSTOMER_ORDER_HISTORY as
select O.ORDER_ID, O.ORDER_DATE, C.CUSTOMER_NAME, P.PRODUCT_NAME, O.SHIPPING
  from ORDER_HISTORY O, CUSTOMER_INFO C, PRODUCT_INFO P
where O.CUSTOMER = C.CUSTOMER_ID and O.PRODUCT = P.PRODUCT_ID;
```

Script Output | Query Result

Task completed in 0.131 seconds

```
view CUSTOMER_ORDER_HISTORY created.
```

Worksheet | Query Builder

```
create or replace view ORDER_SHIP_CALENDAR as
select O.ORDER_DATE, S.SHIP_DATE, SI.DAYS_TO_DELIVERY, S.ESTIMATED_ARRIVAL
  from ORDER_HISTORY O, SHIPPING_HISTORY S, SHIPPING_INFO SI
where O.ORDER_ID = S.ORDER_ID and S.SHIP_TYPE = SI.SHIPPING_TYPE;
```

Script Output | Query Result

Task completed in 0.162 seconds

```
view ORDER_SHIP_CALENDAR created.
```

```
create or replace view PRODUCT_POPULARITY as
select P.PRODUCT_NAME, P.COLOR, COUNT(O.PRODUCT) as PURCHASES
  from PRODUCT_INFO P, ORDER_HISTORY O
where P.PRODUCT_ID (+) = O.PRODUCT
group by P.PRODUCT_NAME, P.COLOR, O.PRODUCT
order by P.PRODUCT_NAME;
```

Script Output

Task completed in 0.18 seconds

```
view PRODUCT_POPULARITY created.
```

```
create or replace view SHIPPING_POPULARITY as
select SI.SHIPPING_TYPE, Count(*) as POPULARITY
  from SHIPPING_INFO SI left outer join SHIPPING_HISTORY S on SI.SHIPPING_TYPE = S.SHIP_TYPE
group by SI.SHIPPING_TYPE;
```

Script Output

Task completed in 0.663 seconds

```
view SHIPPING_POPULARITY created.
```

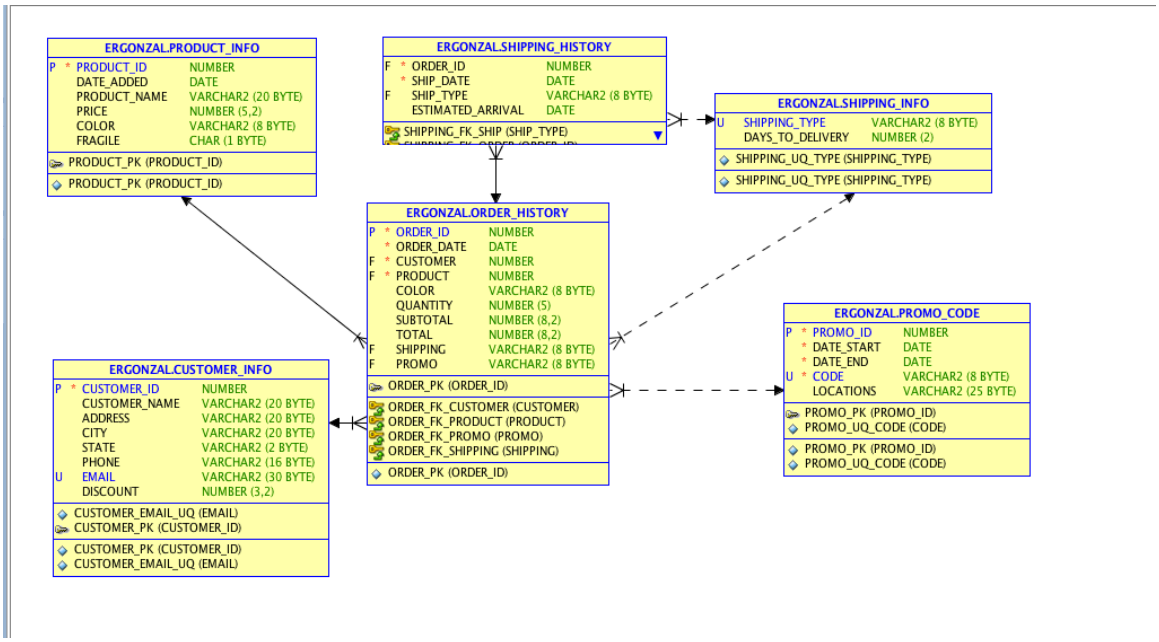
```
create or replace view PROMO_POPULARITY as
select P.CODE, COUNT(*) as POPULARITY
  from PROMO_CODE P, ORDER_HISTORY O
where P.CODE (+) = O.PROMO
group by P.CODE;
```

Script Output

Task completed in 0.116 seconds

```
view PROMO_POPULARITY created.
```

4. E-R Diagram



5. SQL Database Queries
   a. String Functions

Worksheet | Query Builder

```
create or replace view CUSTOMER_ORDER_HISTORY as
select
  CONCAT( 'C', LPAD(O.ORDER_ID, 5, 0)) as ORDER_ID,
  O.ORDER_DATE,
  RPAD(SUBSTR(CUSTOMER_NAME, 0, INSTR(C.CUSTOMER_NAME, ' ') - 1), 10, ' ') as FIRST_NAME,
  RPAD(SUBSTR(CUSTOMER_NAME, INSTR(C.CUSTOMER_NAME, ' ') + 1), 10, ' ') as LAST_NAME,
  P.PRODUCT_NAME,
  O.SHIPPING
  from ORDER_HISTORY O, CUSTOMER_INFO C, PRODUCT_INFO P
where O.CUSTOMER = C.CUSTOMER_ID and O.PRODUCT = P.PRODUCT_ID;
```

Script Output | Query Result

SQL | All Rows Fetched: 11 in 0.062 seconds

	ORDER_ID	ORDER_DATE	FIRST_NAME	LAST_NAME	PRODUCT_NAME	SHIPPING
1	C00000	03-JAN-01	BARRY	ALLEN	WONDER AIR GI	REG
2	C00001	03-JAN-01	BARRY	ALLEN	WONDER AIR GI	REG
3	C00002	03-MAR-03	DICK	GRAYSON	WONDER AIR GI	EXPR
4	C00003	03-MAR-03	DICK	GRAYSON	WONDER ELITE GI	EXPR
5	C00004	03-JUL-03	(null)	DREAM	WONDER ELITE GI	NIGHT
6	C00005	03-JUL-03	(null)	DREAM	WONDER ELITE GI	NIGHT
7	C00006	03-JUL-03	(null)	DEATH	WONDER ELITE GI	NIGHT
8	C00007	23-AUG-10	WALLY	WEST	WONDER BOKKEN	REG
9	C00008	23-AUG-10	WALLY	WEST	WONDER BO STAFF	REG
10	C00009	24-AUG-10	WALLY	WEST	WONDER ELITE GI	REG
11	C00010	24-AUG-13	(null)	DREAM	GALVAO SIGNED GI	NIGHT

b. Number Functions

Worksheet | Query Builder

```
create or replace view PRODUCT_POPULARITY as
select
    P.PRODUCT_ID,
    P.PRODUCT_NAME,
    P.COLOR,
    COUNT(O.PRODUCT) as SALES,
    SUM(O.QUANTITY) as INCOME
  from PRODUCT_INFO P, ORDER_HISTORY O
where P.PRODUCT_ID (+) = O.PRODUCT
group by P.PRODUCT_ID, P.PRODUCT_NAME, P.COLOR, O.PRODUC
order by P.PRODUCT_NAME;
```

Script Output | Query Result

SQL | All Rows Fetched: 8 in 0.06 seconds

	PRODUCT_ID	PRODUCT_NAME	COLOR	SALES	INCOME
1	9	GALVAO SIGNED GI	BLACK	1	1
2	0	WONDER AIR GI	WHITE	1	2
3	1	WONDER AIR GI	BLUE	1	5
4	3	WONDER AIR GI	BLACK	1	1
5	7	WONDER BO STAFF	(null)	1	4
6	6	WONDER BOKKEN	(null)	1	1
7	4	WONDER ELITE GI	WHITE	1	4
8	5	WONDER ELITE GI	BLACK	4	16

c. Date Functions

Worksheet | Query Builder

```
create or replace view PROMO_ORDER_HISTORY as
select
    P.Code,
    CONCAT( 'C', LPAD(O.ORDER_ID, 5, 0)) as ORDER_ID,
    O.ORDER_DATE,
    DECODE( SIGN(MONTHS_BETWEEN(O.ORDER_DATE, P.DATE_START)) *
      SIGN(MONTHS_BETWEEN(P.DATE_END, O.ORDER_DATE)), 1, 'VALID', 'INVALID')
        as VALIDITY
  from PROMO_CODE P, ORDER_HISTORY O
where P.CODE (+) = O.PROMO;
```

Script Output | Query Result

SQL | All Rows Fetched: 11 in 0.071 seconds

	CODE	ORDER_ID	ORDER_DATE	VALIDITY
1	WNDRYRS	C00000	03-JAN-01	VALID
2	WNDRYRS	C00001	03-JAN-01	VALID
3	WNDRBLL	C00002	03-MAR-03	VALID
4	WNDRBLL	C00003	03-MAR-03	VALID
5	WNDRBLL	C00004	03-JUL-03	VALID
6	WNDRBLL	C00005	03-JUL-03	VALID
7	WNDRBLL	C00006	03-JUL-03	VALID
8	WNDRFLL	C00007	23-AUG-10	VALID
9	WNDRFLL	C00008	23-AUG-10	VALID
10	WNDRFLL	C00009	24-AUG-10	VALID
11	(null)	C00010	24-AUG-13	INVALID

d. Decode Function

-- See section 5c or 8 --

6. Use Group By and Having

```
select
    LPAD(SUBSTR(CUSTOMER_NAME, 0, INSTR(C.CUSTOMER_NAME, ' ')), 10, ' ') as FIRST_NAME,
    LPAD(SUBSTR(CUSTOMER_NAME, INSTR(C.CUSTOMER_NAME, ' ') + 1), 10, ' ') as LAST_NAME,
    SUM(O.QUANTITY) as PRODUCTS_PURCHASED
  from CUSTOMER_INFO C left outer join ORDER_HISTORY O on C.CUSTOMER_ID = O.CUSTOMER
group by
  LPAD(SUBSTR(CUSTOMER_NAME, 0, INSTR(C.CUSTOMER_NAME, ' ')), 10, ' '),
  LPAD(SUBSTR(CUSTOMER_NAME, INSTR(C.CUSTOMER_NAME, ' ') + 1), 10, ' ');
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.504 seconds

	FIRST_NAME	LAST_NAME	PRODUCTS_PURCHASED
1	DICK	GRAYSON	9
2	THE	MONITOR	(null)
3	BARRY	ALLEN	3
4	(null)	DEATH	4
5	WALLY	WEST	10
6	JASON	TODD	(null)
7	(null)	DREAM	8

```
select
    LPAD(SUBSTR(CUSTOMER_NAME, 0, INSTR(C.CUSTOMER_NAME, ' ')), 10, ' ') as FIRST
    LPAD(SUBSTR(CUSTOMER_NAME, INSTR(C.CUSTOMER_NAME, ' ') + 1), 10, ' ') as LAST
    SUM(O.QUANTITY) as PRODUCTS_PURCHASED
  from CUSTOMER_INFO C left outer join ORDER_HISTORY O on C.CUSTOMER_ID = O.CUSTO
group by
  LPAD(SUBSTR(CUSTOMER_NAME, 0, INSTR(C.CUSTOMER_NAME, ' ')), 10, ' '),
  LPAD(SUBSTR(CUSTOMER_NAME, INSTR(C.CUSTOMER_NAME, ' ') + 1), 10, ' ')
having
  SUM(QUANTITY) > 0;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.063 seconds

	FIRST_NAME	LAST_NAME	PRODUCTS_PURCHASED
1	DICK	GRAYSON	9
2	BARRY	ALLEN	3
3	(null)	DEATH	4
4	WALLY	WEST	10
5	(null)	DREAM	8

7. Use Subqueries

```
select
    distinct LPAD(SUBSTR(C.CUSTOMER_NAME, 0, INSTR(C.CUSTOMER_NAME, ' ')), 10, ' ') as FIRST_NAME,
    LPAD(SUBSTR(C.CUSTOMER_NAME, INSTR(C.CUSTOMER_NAME, ' ') + 1), 10, ' ') as LAST_NAME
  from CUSTOMER_INFO C
where C.CUSTOMER_ID in
    ( select CUSTOMER from ORDER_HISTORY where COLOR = 'BLACK')
  and C.CUSTOMER_ID not in
    (select customer from ORDER_HISTORY where ORDER_DATE > TO_DATE('01-JAN-03'));
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.126 seconds

	FIRST_NAME	LAST_NAME
1	BARRY	ALLEN

8. Use Decode

```
select
    distinct CONCAT( 'C', LPAD( O.ORDER_ID, 5, 0)) as ORDER_ID,
    LPAD(DECODE(
      ROUND((P.Price * O.Quantity * (1 - C.DISCOUNT) * DECODE(POH.VALIDITY, 'VALID', .9, 1)), 2),
      0, 'The Endless',
      ROUND((P.Price * O.Quantity * (1 - C.DISCOUNT) * DECODE(POH.VALIDITY, 'VALID', .9, 1)), 2)), 15, ' ')
      as SUBTOTAL
  from ORDER_HISTORY O, PRODUCT_INFO P, CUSTOMER_INFO C, PROMO_ORDER_HISTORY POH
where O.PRODUCT = P.PRODUCT_ID and O.CUSTOMER = C.CUSTOMER_ID and O.ORDER_DATE = POH.ORDER_DATE
order by ORDER_ID;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 11 in 0.088 seconds

	ORDER_ID	SUBTOTAL
1	C00000	156.89
2	C00001	92.84
3	C00002	267.64
4	C00003	359.91
5	C00004	The Endless
6	C00005	The Endless
7	C00006	The Endless
8	C00007	30.56
9	C00008	122.25
10	C00009	764.81
11	C00010	The Endless

9. Use Join and Outer Join
    a. Inner join

```
select
    C.Customer_NAME, O.ORDER_DATE
  from CUSTOMER_INFO C inner join ORDER_HISTORY O on C.CUSTOMER_ID = O.CUSTOMER;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 11 in 0.338 seconds

	CUSTOMER_NAME	ORDER_DATE
1	BARRY ALLEN	03-JAN-01
2	BARRY ALLEN	03-JAN-01
3	DICK GRAYSON	03-MAR-03
4	DICK GRAYSON	03-MAR-03
5	DREAM	03-JUL-03
6	DREAM	03-JUL-03
7	DEATH	03-JUL-03
8	WALLY WEST	23-AUG-10
9	WALLY WEST	23-AUG-10
10	WALLY WEST	24-AUG-10
11	DREAM	24-AUG-13

    b. Outer join

```
--outer join --
select
    C.Customer_NAME, O.ORDER_DATE
  from CUSTOMER_INFO C left outer join ORDER_HISTORY O on C.CUSTOMER_ID = O.CUSTOMER;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 13 in 2.341 seconds

	CUSTOMER_NAME	ORDER_DATE
1	BARRY ALLEN	03-JAN-01
2	BARRY ALLEN	03-JAN-01
3	WALLY WEST	23-AUG-10
4	WALLY WEST	23-AUG-10
5	WALLY WEST	24-AUG-10
6	DICK GRAYSON	03-MAR-03
7	DICK GRAYSON	03-MAR-03
8	JASON TODD	(null)
9	THE MONITOR	(null)
10	DREAM	24-AUG-13
11	DREAM	03-JUL-03
12	DREAM	03-JUL-03
13	DEATH	03-JUL-03

10. Demonstrate Savepoints

```
-- Step 10 : Savepoints --
Savepoint Step10;

insert into CUSTOMER_INFO (CUSTOMER_ID, CUSTOMER_NAME, ADDRESS, EMAIL, DISCOUNT)
  values (
  (select Count(*) from CUSTOMER_INFO) + 1, 'DELIGHT', 'HAPPINESS', 'SMILEYFISH@GMAIL.COM', 1);

Savepoint Step10b;

insert into CUSTOMER_INFO (CUSTOMER_ID, CUSTOMER_NAME, ADDRESS, EMAIL, DISCOUNT)
  values (
  (select Count(*) from CUSTOMER_INFO) + 1, 'DELIRIUM', 'COLORS', 'RAINBOWFISH@GMAIL.COM', 1);

rollback to Step10b;

rollback to Step10;

insert into CUSTOMER_INFO (CUSTOMER_ID, CUSTOMER_NAME, ADDRESS, EMAIL, DISCOUNT)
  values (
  (select Count(*) from CUSTOMER_INFO) + 1, 'DESTRUCTION', 'CRETE', 'BARNABAS@GMAIL.COM', 1);

commit;
```

Script Output × | Query Result ×

Task completed in 0.054 seconds

```
Savepoint Step10
1 rows inserted.
Savepoint Step10b
1 rows inserted.
rollback complete.
rollback complete.
1 rows inserted.
committed.
```

11. Insert Statements

```
create table ENDLESS_PURCHASES (
  ORDER_DATE        DATE,
  ENDLESS_NAME      VARCHAR2(20),
  ITEM              VARCHAR2(20),
  QUANTITY          NUMBER,
  SUBTOTAL          VARCHAR2(45)
);

create table HUMAN_PURCHASES (
  ORDER_DATE        DATE,
  HUMAN_NAME        VARCHAR2(20),
  ITEM              VARCHAR2(20),
  QUANTITY          NUMBER,
  SUBTOTAL          VARCHAR2(45)
);

create table BIG_SPENDERS (
  ORDER_DATE        DATE,
  INDIVIDUAL_NAME   VARCHAR2(20),
  ITEM              VARCHAR2(20),
  QUANTITY          NUMBER,
  SUBTOTAL          VARCHAR2(45)
);
```

Script Output | Query Result | Task completed in 0.183 seco

```
table ENDLESS_PURCHASES created.
table HUMAN_PURCHASES created.
table BIG_SPENDERS created.
```

a. Insert All

```
--inserts 4 rows in endless_purchases, 2 rows into human_purchases, 9 rows into big_spenders--
insert ALL
    when LENGTH(SUBTOTAL) > 8 then
      into ENDLESS_PURCHASES
    when LENGTH(SUBTOTAL) > 5 then
      into BIG_SPENDERS
    else
      into HUMAN_PURCHASES
select COH.ORDER_DATE, TRIM(' ' from COH.LAST_NAME) as LAST_NAME, COH.PRODUCT_NAME, OH.QUANTITY, TRIM(' ' from OS.SUBTOTAL) as SUBTOTAL
  from CUSTOMER_ORDER_HISTORY COH, ORDER_SUBTOTAL OS, ORDER_HISTORY OH
where
    (COH.ORDER_ID = OS.ORDER_ID
      and CONCAT( 'C', LPAD( OH.ORDER_ID, 5, 0)) = COH.ORDER_ID);


delete ENDLESS_PURCHASES;
delete HUMAN_PURCHASES;
delete BIG_SPENDERS;
```

Script Output | Query Result | Task completed in 0.479 seconds

```
table ENDLESS_PURCHASES created.
table HUMAN_PURCHASES created.
table BIG_SPENDERS created.
15 rows inserted.
4 rows deleted.
2 rows deleted.
9 rows deleted.
```

b. Insert First

```
--filters the endless out of the big_spenders, since they don't pay anything --
--inserts 4 rows into endless_purchases, 5 rows into big_spenders, 2 into human_purchases --
insert FIRST
    when LENGTH(SUBTOTAL) > 8 then
      into ENDLESS_PURCHASES
    when LENGTH(SUBTOTAL) > 5 then
      into BIG_SPENDERS
    else
      into HUMAN_PURCHASES
select COH.ORDER_DATE, TRIM(' ' from COH.LAST_NAME) as LAST_NAME, COH.PRODUCT_NAME, OH.QUANTITY, TRIM(' ' from OS.SUBTOTAL) as SUBTOTAL
  from CUSTOMER_ORDER_HISTORY COH, ORDER_SUBTOTAL OS, ORDER_HISTORY OH
where
    (COH.ORDER_ID = OS.ORDER_ID
      and CONCAT( 'C', LPAD( OH.ORDER_ID, 5, 0)) = COH.ORDER_ID);

drop table ENDLESS_PURCHASES;
drop table HUMAN_PURCHASES;
drop table BIG_SPENDERS;

-- Step 12 : Update and Embedded Select --
```

Script Output | Query Result

Task completed in 0.191 seconds

```
4 rows deleted.
2 rows deleted.
9 rows deleted.
11 rows inserted.
table ENDLESS_PURCHASES dropped.
table HUMAN_PURCHASES dropped.
table BIG_SPENDERS dropped.
```

12. Demonstrate Update and Embedded Select


```
--filters the endless out of the big_spenders, since they don't pay anything --
--inserts 4 rows into endless_purchases, 5 rows into big_spenders, 2 into human_purchases --
insert FIRST
    when LENGTH(SUBTOTAL) > 8 then
      into ENDLESS_PURCHASES
    when LENGTH(SUBTOTAL) > 5 then
      into BIG_SPENDERS
    else
      into HUMAN_PURCHASES
select COH.ORDER_DATE, TRIM(' ' from COH.LAST_NAME) as LAST_NAME, COH.PRODUCT_NAME, OH.QUANTITY, TRIM(' ' from OS.SUBTOTAL) as SUBTOTAL
  from CUSTOMER_ORDER_HISTORY COH, ORDER_SUBTOTAL OS, ORDER_HISTORY OH
where
    (COH.ORDER_ID = OS.ORDER_ID
      and CONCAT( 'C', LPAD( OH.ORDER_ID, 5, 0)) = COH.ORDER_ID);

drop table ENDLESS_PURCHASES;
drop table HUMAN_PURCHASES;
drop table BIG_SPENDERS;

-- Step 12 : Update and Embedded Select --
```

Script Output | Query Result

Task completed in 0.191 seconds

```
4 rows deleted.
2 rows deleted.
9 rows deleted.
11 rows inserted.
table ENDLESS_PURCHASES dropped.
table HUMAN_PURCHASES dropped.
table BIG_SPENDERS dropped.
```


13. Demonstrate Merge

```
-- Step 13 : Merge --
savepoint Step13;

-- change Order_History so can accept VARCHAR2 types in the SUBTOTAL column
alter table ORDER_HISTORY modify (SUBTOTAL VARCHAR2(45));

-- merge all 11 rows of subtotal column from view ORDER_SUBTOTAL into ORDER_HISTORY --
merge into ORDER_HISTORY OH
  using (select ORDER_ID, SUBTOTAL from ORDER_SUBTOTAL) OS
    on ((CONCAT( 'C', LPAD( OH.ORDER_ID, 5, 0)) = OS.ORDER_ID))
  when matched then
    update set SUBTOTAL = OS.SUBTOTAL;
```

Script Output | Query Result

Task completed in 0.181 seconds

```
table ORDER_HISTORY altered.
11 rows merged.
```

14. Create a unique constraint

```
-- Step 14 : Create a Unique constraint --
-- accomplished in step 2 while creating tables CUSTOMER_INFO, SHIPPING_INFO, PROMO_CODE--
```

15. Create a table from another table

```
-- Step 15 : Create a table from another table --
create table DAILY_PROFIT as
select ORDER_DATE, SUM( DECODE(TRIM(' ' from SUBTOTAL), 'The Endless', 0, SUBTOTAL)) as DAILY_SUBTOTAL
  from ORDER_HISTORY
group by ORDER_DATE
order by ORDER_DATE;
```

Script Output | Query Result

Task completed in 0.116 seconds

```
table DAILY_PROFIT created.
```

16. Create an index

```
-- Step 16 : Create an index --
savepoint Step16;
create index DAILY_NDX on DAILY_PROFIT(ORDER_DATE);
create unique index CUSTOMER_NAME_NDX on CUSTOMER_INFO(CUSTOMER_NAME);
```

Script Output | Query Result

Task completed in 0.095 seconds

```
index DAILY_NDX created.
unique index CUSTOMER_NAME_NDX created.
```

17. Create a sequence

```
-- Step 17 : Create an sequence --
-- can access and easily increment for next ID of PRODUCT_INFO and CUSTOMER_INFO--
-- note : accidently skipped PRODUCT_ID = 2 during initial set-up --
create sequence PRODUCT_ID_SEQ increment by 1 start with 9;
create sequence CUSTOMER_ID_SEQ increment by 1 start with 9;
```

Script Output | Query Result

Task completed in 0.085 seconds

```
sequence PRODUCT_ID_SEQ created.
sequence CUSTOMER_ID_SEQ created.
```

18. Materialized views and materialized view logs

```oracle_plsql
-- Step 18 : Materialized Views --
-- 18a : Create --
-- creates local version of customer_info table, refreshes every day --
create materialized view LOCAL_CUSTOMER
  refresh force
    start with SYSDATE next SYSDATE+1
  as
    select * from CUSTOMER_INFO;

-- 18b : Drop --
drop materialized view LOCAL_CUSTOMER;

-- 18c : Create log --
create materialized view log on CUSTOMER_INFO
with sequence
including new values;

-- 18d : Drop log --

drop materialized view log on CUSTOMER_INFO;

-- 18e : Manually refresh view --

execute DBMS_MVIEW.REFRESH('LOCAL_CUSTOMER', '?');

-- 18f : Modify view --

alter materialized view LOCAL_CUSTOMER disable query rewrite;

-- Step 19 : Create Triggers --

-- Step 20 : Create Stored Procedures --
```

Script Output x | Query Result x

Task completed in 0.296 seconds

```
materialized view LOCAL_CUSTOMER created.
materialized view LOCAL_CUSTOMER dropped.
materialized view LOG created.
materialized view LOG dropped.
materialized view LOCAL_CUSTOMER created.
anonymous block completed
materialized view LOCAL_CUSTOMER altered.
materialized view LOCAL_CUSTOMER dropped.
```

19. Create Triggers

```oracle_plsql
-- Step 19 : Create Triggers --
-- should keep a second set of records for order_history but changes all values of canceled
-- orders to (canceled) but still available to read and edit. keeps customer_id for remarketing--

create table ORDER_HISTORY_BACKUP as
select * from ORDER_HISTORY;


--trigger compiled when highlighted from set define off; to end;
set define off;
create or replace trigger ORDER_BACKUP_TRG
  before insert or update or delete on ORDER_HISTORY
for each row
declare
  del_note VARCHAR2(10);
begin
  del_note := '(canceled)';
  if INSERTING then
    insert into ORDER_HISTORY_BACKUP (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, COLOR, QUANTITY, SHIPPING, PROMO)
      values (ORDER_ID_SEQ.nextVal, :new.ORDER_DATE, :new.CUSTOMER, :new.PRODUCT,
        :new.COLOR, :new.QUANTITY, :new.SHIPPING, :new.PROMO);
  end if;
  if UPDATING then
    update ORDER_HISTORY_BACKUP set CUSTOMER = :new.CUSTOMER, PRODUCT = :new.PRODUCT, color = :new.COLOR,
      QUANTITY = :new.QUANTITY, SHIPPING = :new.SHIPPING, PROMO = :new.PROMO
    where ORDER_ID = :new.ORDER_ID and ORDER_DATE = :new.ORDER_DATE;
  else
    update ORDER_HISTORY_BACKUP set PRODUCT = del_note, color = del_note,
      QUANTITY = del_note, SHIPPING = del_note, PROMO = del_note
    where ORDER_ID = :new.ORDER_ID and ORDER_DATE = :new.ORDER_DATE;
  end if;
end;
/
```

Script Output | Query Result

Task completed in 0.201 seconds

```
TRIGGER ORDER_BACKUP_TRG compiled
```

```oracle_plsql
create or replace trigger PRODUCT_INFO_TRG
  after insert on PRODUCT_INFO
for each row
begin
  update PRODUCT_INFO set PRODUCT_ID = PRODUCT_ID_SEQ.nextVal;
end;
/

create or replace trigger CUSTOMER_INFO_TRG
  after insert on CUSTOMER_INFO
for each row
begin
  update CUSTOMER_INFO set CUSTOMER_ID = CUSTOMER_ID_SEQ.nextVal;
end;
```

Script Output | Query Result

Task completed in 0.125 seconds

```
TRIGGER ORDER_BACKUP_TRG compiled
TRIGGER PRODUCT_INFO_TRG compiled
TRIGGER CUSTOMER_INFO_TRG compiled
```

20. Create stored procedures

```oracle_table_ref
-- Step 20 : Create Stored Procedures --
--looks up a customer_id given a customer_name
create or replace function CUSTOMER_ID_LOOKUP (aName in VARCHAR2)
  return NUMBER
  is
  found_id NUMBER;
begin
  select CUSTOMER_ID into found_id
  from CUSTOMER_INFO where CUSTOMER_NAME = aName;
  return (found_id);
end;
/

--looks up a customer_name given a customer id--
create or replace function CUSTOMER_NAME_LOOKUP (anID in NUMBER)
  return VARCHAR2
  is
  found_name VARCHAR2(20);
begin
  select CUSTOMER_NAME into found_name
  from CUSTOMER_INFO where CUSTOMER_ID = anID;
  return (found_name);
end;
/

--looks up a discount rate given a customer_id --
create or replace function DISCOUNT_LOOKUP (anID in NUMBER)
  return NUMBER
  is
  found_discount NUMBER(3,2);
begin
  select (1 - DISCOUNT) into found_discount
  from CUSTOMER_INFO where CUSTOMER_ID = anID;
  return (found_discount);
end;
/

--looks up a product_id given a product_name and a color--
create or replace function PRODUCT_ID_LOOKUP (aName in VARCHAR2, aColor in VARCHAR2)
  return NUMBER
  is
  found_id NUMBER;
begin
  select PRODUCT_ID into found_id
  from PRODUCT_INFO where PRODUCT_NAME = aName and COLOR = aCOLOR;
  return (found_id);
end;
/
```

```oracle
--looks up price from product_info, given a product_id --
create or replace function PRICE_LOOKUP (aProductID in NUMBER)
  return NUMBER
  is
  found_price NUMBER(5,2);
begin
  select PRICE into found_price
  from PRODUCT_INFO where PRODUCT_ID = aProductID;
  return (found_price);
end;
/

-- checks if in promo_code table--
create or replace function VALID_CHECk (aCode in VARCHAR2)
  return NUMBER
  is
  validity NUMBER(3, 2);
begin
  select count(*) into validity from PROMO_CODE where CODE = aCODE;
  return validity;
end;
/

--checks if in promo_code table and then checks if still valid--
create or replace function PROMO_CALC (aCode in VARCHAR2, aDate in DATE)
  return NUMBER
  is
  validity NUMBER(3,2);
begin
   if VALID_CHECK(aCode) > 0 then
    select DECODE( SIGN(MONTHS_BETWEEN(aDATE, P.DATE_START)) *
        SIGN(MONTHS_BETWEEN(P.DATE_END, aDATE)), 1, .9, 1) into validity
      from PROMO_CODE P where P.CODE = aCode;
   else
    validity := 0;
  end if;
  return validity;
end;
/
```

```oracle
--calcuates subtotal--
create or replace function CALC_SUBTOTAL (aDate in DATE, aCode in VARCHAR2, aCustomerID in NUMBER,
    aProductID in NUMBER, aQuantity NUMBER)
  return VARCHAR2
  as
  subtotal VARCHAR2(45);
begin
  select
  LPAD(
      DECODE(
        (aQuantity * PRICE_LOOKUP(aProductID) * DISCOUNT_LOOKUP(aCUSTOMERID) * PROMO_CALC(aCode, aDATE)),
        0,
        'The Endless',
        ROUND((aQuantity * PRICE_LOOKUP(aProductID) * DISCOUNT_LOOKUP(aCUSTOMERID) * PROMO_CALC(aCode, aDATE)
      ),
    15, ' ')
    into subtotal
  from Dual;
  return (subtotal);
end;
/

--calculates the total--
create or replace function CALC_TOTAL (aSub in VARCHAR2)
  return VARCHAR2
  as
  total VARCHAR2(45);
begin
  select LPAD(Round(Decode(TRIM(' ' from aSub), 'The Endless', 0, aSub * 1.07), 2), 15, ' ') into total
    from Dual;
  return total;
end;
/
```

```oracle
-- uses above functions to create a new order given limited information --
create or replace procedure NEW_ORDER (ORDER_DATE in DATE,
    NEW_NAME in VARCHAR2, PRODUCT_NAME in VARCHAR2, COLOR in VARCHAR2,
    QUANTITY in NUMBER, SHIPPING in VARCHAR2, PROMO in VARCHAR2)
as
begin
  insert into ORDER_HISTORY (ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, COLOR, QUANTITY, SUBTOTAL, TOTAL, SHIPPING, PROMO)
    values
      (ORDER_ID_SEQ.NextVal,
        ORDER_DATE,
        CUSTOMER_ID_LOOKUP(NEW_NAME),
        PRODUCT_ID_LOOKUP(PRODUCT_NAME, COLOR),
        COLOR,
        QUANTITY,
        CALC_SUBTOTAL(ORDER_DATE, PROMO, CUSTOMER_ID_LOOKUP(NEW_NAME), PRODUCT_ID_LOOKUP(PRODUCT_NAME, COLOR), QUANTITY),
        CALC_TOTAL(CALC_SUBTOTAL(ORDER_DATE, PROMO, CUSTOMER_ID_LOOKUP(NEW_NAME), PRODUCT_ID_LOOKUP(PRODUCT_NAME, COLOR), QUANTITY)),
        SHIPPING,
        DECODE(VALID_CHECK(PROMO), 0, (null), PROMO));
end;
/

execute NEW_ORDER(TO_DATE('11-AUG-14'), 'DESTRUCTION', 'WONDER ELITE GI', 'BLACK', 3, 'REG', 'DOG');
```

21. Create objects in database

```oracle
--create product object containing information most often needed --
create or replace type PRODUCT_TY as object
( P_ID     NUMBER,
  P_NAME   VARCHAR2(20),
  PRICE    NUMBER(5,2),
  COLOR    VARCHAR2(8));
/

create table PRODUCTS
(DATE_ADDED    DATE,
 PRODUCT       PRODUCT_TY);


--populate new table and build mulitple product_ty objects--
insert all
  into PRODUCTS
    values (DATE_ADDED, PRODUCT_TY(PRODUCT_ID, PRODUCT_NAME, PRICE, COLOR))
select DATE_ADDED, PRODUCT_ID, PRODUCT_NAME, PRICE, COLOR
  from PRODUCT_INFO;
```

```oracle
--populate new table and build mulitple product_ty objects--
insert all
  into PRODUCTS
    values (DATE_ADDED, PRODUCT_TY(PRODUCT_ID, PRODUCT_NAME, PRICE, COLOR))
select DATE_ADDED, PRODUCT_ID, PRODUCT_NAME, PRICE, COLOR
  from PRODUCT_INFO;

-- create order objects containing information most-often accessed --
create or replace type ORDERS_TY as object
( O_Id     NUMBER,
  O_Date   DATE,
  O_Cust   NUMBER,
  O_Prod   NUMBER,
  O_Sub    VARCHAR2(45),
  O_Tot    VARCHAR2(45),
  O_Pro    VARCHAR2(8)
);
/

create table ORDER_TAB
(Orders    ORDERS_TY);

--populate the order_tab table with orders_ty objects made from orders in order_history
insert all
  into ORDER_TAB
    values ( ORDERS_TY(ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SUBTOTAL, TOTAL, PROMO))
select ORDER_ID, ORDER_DATE, CUSTOMER, PRODUCT, SUBTOTAL, TOTAL, PROMO
  from ORDER_HISTORY;
```

Script Output | Query Result

Task completed in 0.085 seconds

```
TYPE PRODUCT_TY compiled
table PRODUCTS created.
9 rows inserted.
TYPE ORDERS_TY compiled
table ORDER_TAB created.
11 rows inserted.
```