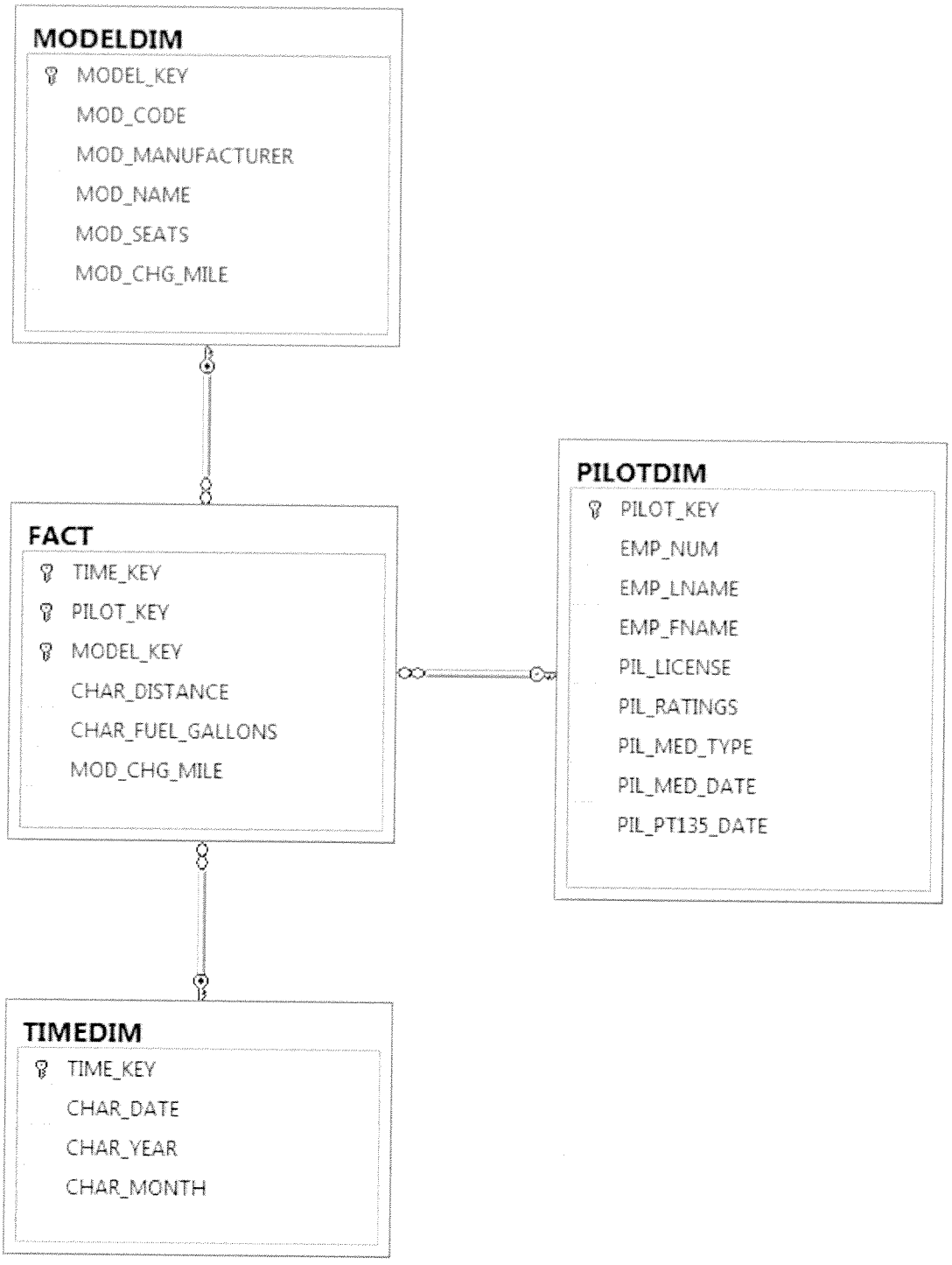
SOLUTION TO ASSIGNMENT 12

PART 2



PART 3

```
--CREATING THE TABLES FOR THE DATA WAREHOUSE
--THESE DROP TABLE STATEMENTS CAN BE USED TO DROP ALL THE TABLES SO THEY CAN BE RECREATED
IN CASE THE TABLES NEED TO BE RECREATED

DROP TABLE FACT
DROP TABLE TIMEDIM
DROP TABLE PILOTDIM
DROP TABLE MODELDIM
DROP TABLE STAGING

CREATE TABLE TIMEDIM (
TIME_KEY INT IDENTITY NOT NULL,
CHAR_DATE datetime,
CHAR_YEAR INT,
CHAR_MONTH INT
);


CREATE TABLE MODELDIM (
MODEL_KEY INT IDENTITY NOT NULL,
MOD_CODE varchar(10),
MOD_MANUFACTURER varchar(15),
MOD_NAME varchar(20),
MOD_SEATS float(8),
MOD_CHG_MILE numeric
);


CREATE TABLE PILOTDIM (
PILOT_KEY INT IDENTITY NOT NULL,
EMP_NUM int,
EMP_LNAME varchar(15),
EMP_FNAME varchar(15),
PIL_LICENSE varchar(25),
PIL_RATINGS varchar(30),
PIL_MED_TYPE varchar(1),
PIL_MED_DATE datetime,
PIL_PT135_DATE datetime
);

CREATE TABLE FACT (
TIME_KEY INT NOT NULL,
PILOT_KEY INT NOT NULL,
MODEL_KEY INT NOT NULL,
CHAR_DISTANCE float(8),
CHAR_FUEL_GALLONS float(8),
MOD_CHG_MILE numeric
)

--CREATE THE CONSTRAINTS
ALTER TABLE PILOTDIM
ADD CONSTRAINT PK_PILOTDIM PRIMARY KEY (PILOT_KEY)

ALTER TABLE TIMEDIM
```

```
ADD CONSTRAINT PK_TIMEDIM PRIMARY KEY (TIME_KEY)

ALTER TABLE MODELDIM
ADD CONSTRAINT PK_MODELDIM PRIMARY KEY (MODEL_KEY)

ALTER TABLE FACT
ADD    CONSTRAINT PK_FACT PRIMARY KEY (PILOT_KEY, TIME_KEY, MODEL_KEY),
       CONSTRAINT FK_FACT_PILOTDIM FOREIGN KEY (PILOT_KEY) REFERENCES PILOTDIM,
       CONSTRAINT FK_FACT_TIMEDIM FOREIGN KEY (TIME_KEY) REFERENCES TIMEDIM,
       CONSTRAINT FK_FACT_MODELDIM FOREIGN KEY (MODEL_KEY) REFERENCES MODELDIM


--CREATE THE STAGING TABLE
CREATE TABLE STAGING (
TIME_KEY INT,
PILOT_KEY INT,
MODEL_KEY INT,
EMP_NUM int,
MOD_CODE varchar(10),
CHAR_DATE datetime,
CHAR_DISTANCE float(8),
CHAR_FUEL_GALLONS float(8),
MOD_CHG_MILE numeric
)
```

PART 4

```
ALTER PROCEDURE A12
AS
BEGIN
       --FIRST DISABLE THE CONSTRAINTS SO THE TABLES CAN BE TRUNCATED
       --TRUNCATING THE TABLES ENABLES US TO REPOPULATE THE DATA WAREHOUSE TABLES
       --WE WANT TO BE ABLE TO REPOPULATE BECAUSE DEVELOPMENT AND TESTING ALWAYS TAKE
MANY TRIES

       ALTER TABLE FACT
       DROP CONSTRAINT FK_FACT_PILOTDIM, FK_FACT_TIMEDIM, FK_FACT_MODELDIM

       --TRUNCATE THE TABLES
       TRUNCATE TABLE FACT
       TRUNCATE TABLE PILOTDIM
       TRUNCATE TABLE TIMEDIM
       TRUNCATE TABLE MODELDIM
       TRUNCATE TABLE STAGING

       --PUT THE FK CONSTRAINTS BACK ON
       ALTER TABLE FACT
       ADD    CONSTRAINT FK_FACT_PILOTDIM FOREIGN KEY (PILOT_KEY) REFERENCES PILOTDIM,
              CONSTRAINT FK_FACT_TIMEDIM FOREIGN KEY (TIME_KEY) REFERENCES TIMEDIM,
              CONSTRAINT FK_FACT_MODELDIM FOREIGN KEY (MODEL_KEY) REFERENCES MODELDIM

       --NOW WE ARE READY TO POPULATE THE DIM TABLES
       INSERT INTO PILOTDIM
       SELECT E.EMP_NUM, E.EMP_LNAME, E.EMP_FNAME, P.PIL_LICENSE, P.PIL_RATINGS,
P.PIL_MED_TYPE, P.PIL_MED_DATE, P.PIL_PT135_DATE
       FROM   EMPLOYEE E INNER JOIN PILOT P ON E.EMP_NUM = P.EMP_NUM

       INSERT INTO TIMEDIM
```

```
        SELECT DISTINCT CHAR_DATE, YEAR(CHAR_DATE), MONTH(CHAR_DATE)
        FROM   CHARTER

        INSERT INTO MODELDIM
        SELECT MOD_CODE, MOD_MANUFACTURER, MOD_NAME, MOD_SEATS, MOD_CHG_MILE
        FROM   MODEL

        --INSERT INTO THE STAGING TABLE. PLEASE NOTE YOU DON'T WANT TO JOIN WITH THE
DIMENSION TABLES BECUASE
        --THAT CAN LEAD TO A HUGE PRODUCT. IN MORE REAL SITUATIONS YOU WILL HAVE VERY
COMPLICATED BUSINESS LOGIC YOU
        --HAVE TO EXECUTE SO AT LEAST ONE STAGING TABLE IS USED. YOU NEED TO UNDERSTAND
THE IMPORTANCE OF
        --THE STAGING TABLE.
        INSERT INTO STAGING (EMP_NUM, MOD_CODE, CHAR_DATE, CHAR_DISTANCE,
CHAR_FUEL_GALLONS, MOD_CHG_MILE)
        SELECT P.EMP_NUM, M.MOD_CODE, C.CHAR_DATE, C.CHAR_DISTANCE, C.CHAR_FUEL_GALLONS,
M.MOD_CHG_MILE
        FROM   CHARTER C INNER JOIN PILOT P ON C.CHAR_PILOT = P.EMP_NUM
                         INNER JOIN AIRCRAFT A ON C.AC_NUMBER = A.AC_NUMBER
                         INNER JOIN MODEL M ON A.MOD_CODE = M.MOD_CODE

        --AT THIS POINT THE STAGING TABLE DOES NOT HAVE THE DATA WARESHOUSE KEYS AND THE
FOLLOWING UPDATE STATEMENTS ARE
        --USED TO PROVIDE THAT. BUT THAT DATA WAREHOUSE KEY IS FOUND IN THE DIMENSION
TABLES. SO THE NEXT STEP IS TO MAP
        --THESE DATA WAREHOUSE KEYS IN THE DIMENSION TABLES TO THE FACTS IN THE STAGING
TABLE USING THE COMMON LINK BETWEEN
        --THE DIMENSION TABLES AND THE STAGING TABLE: PRODUCTION KEYS.

        UPDATE STAGING
        SET           TIME_KEY = T.TIME_KEY
        FROM   STAGING S INNER JOIN TIMEDIM T ON S.CHAR_DATE = T.CHAR_DATE

        UPDATE STAGING
        SET           PILOT_KEY = P.PILOT_KEY
        FROM   STAGING S INNER JOIN PILOTDIM P ON S.EMP_NUM = P.EMP_NUM

        UPDATE STAGING
        SET           MODEL_KEY = M.MODEL_KEY
        FROM   STAGING S INNER JOIN MODELDIM M ON S.MOD_CODE = M.MOD_CODE

        --NOW WE HAVE EVERYTHING WE NEED IN THE STAGING TO POPULATE THE FACT TABLE. IT IS
JUST A MATTER OF
        --COPYING IT OVER TO THE FACT TABLE FROM THE STAGING

        INSERT INTO FACT (TIME_KEY, MODEL_KEY, PILOT_KEY, CHAR_DISTANCE,
CHAR_FUEL_GALLONS, MOD_CHG_MILE)
        SELECT TIME_KEY, MODEL_KEY, PILOT_KEY, CHAR_DISTANCE, CHAR_FUEL_GALLONS,
MOD_CHG_MILE
        FROM   STAGING

        --A TRUE GIFT
END
GO
```

PART 5

What is the SQL command to list the total sales by customer and by product, with subtotals by customer and a grand total for all product sales?

```
SELECT      CUS_CODE, P_CODE, SUM(SALE_UNITS*SALE_PRICE) AS TOTSALES
FROM        DWDAYSALESFACT
GROUP BY    CUS_CODE, P_CODE WITH ROLLUP;
```

6. **What is the SQL command to list the total sales by customer, month and product, with subtotals by customer and by month and a grand total for all product sales?**

```
SELECT      CUS_CODE, TM_MONTH, P_CODE, SUM(SALE_UNITS*SALE_PRICE)
            AS TOTSALES
FROM        DWDAYSALESFACT S INNER JOIN DWTIME T ON S.TM_ID = T.TM_ID
GROUP BY    CUS_CODE, TM_MONTH, P_CODE WITH ROLLUP;
```

7. **What is the SQL command to list the total sales by region and customer, with subtotals by region and a grand total for all sales?**

```
SELECT      REG_ID, CUS_CODE, SUM(SALE_UNITS*SALE_PRICE) AS TOTSALES
FROM        DWDAYSALESFACT S INNER JOIN DWCUSTOMER C
            ON S.CUS_CODE = C.CUS_CODE
GROUP BY    REG_ID, CUS_CODE WITH ROLLUP;
```

8. **What is the SQL command to list the total sales by month and product category, with subtotals by month and a grand total for all sales?**

```
SELECT      TM_MONTH, P_CATEGORY, SUM(SALE_UNITS*SALE_PRICE)
            AS TOTSALES
FROM        DWDAYSALESFACT S INNER JOIN DWPRODUCT P ON S.P_CODE = P.P_CODE
            INNER JOIN DWTIME T ON S.TM_ID = T.TM_ID
GROUP BY    TM_MONTH, P_CATEGORY WITH ROLLUP;
```

9. **What is the SQL command to list the number of product sales (number of rows) and total sales by month, with subtotals by month and a grand total for all sales?**

```
SELECT      TM_MONTH, COUNT(*) AS NUMPROD, SUM(SALE_UNITS*SALE_PRICE)
    AS TOTSALES
FROM        DWDAYSALESFACT S INNER JOIN DWTIME T ON S.TM_ID = T.TM_ID
GROUP BY    TM_MONTH WITH ROLLUP;
```

12. **Using the answer to Problem 10 as your base, what command would you need to generate the same output but with subtotals in all columns? (*Hint*: Use the CUBE command).**

```
SELECT      TM_MONTH, P_CATEGORY, COUNT(*) AS NUMPROD,
            SUM(SALE_UNITS*SALE_PRICE) AS TOTSALES
FROM        DWDAYSALESFACT S INNER JOIN DWPRODUCT P ON S.P_CODE = P.P_CODE
            INNER JOIN DWTIME T ON S.TM_ID = T.TM_ID
GROUP BY    TM_MONTH, P_CATEGORY WITH CUBE;
```