US Flights Case Study

Scenario

A client has provided you with a flat file export containing data about domestic flights in the United States. The analytics consultant that you are working with needs the file loaded into a database and appropriately modeled for use with business intelligence tools. In this case, that tool is Tableau.

Case Requirements

1. Loaded the provided data into the PostgreSQL instance using the provided credentials as follows:

psql --host iw-recruiting-test.cygkjm9anrym.us-west-2.rds.amazonaws.com/ --port 5432 --username candidate7522 --dbname tests_data_engineering

Created a master table FLIGHTS with with unprocessed data and junk values:

```
CREATE TABLE flights (
        TRANSACTIONID INTEGER PRIMARY KEY,
        FLIGHTDATE DATE,
        AIRLINECODE TEXT,
        AIRLINENAME TEXT,
        TAILNUM TEXT,
        FLIGHTNUM INTEGER,
        ORIGINAIRPORTCODE TEXT,
        ORIGAIRPORTNAME TEXT,
        ORIGINCITYNAME TEXT,
        ORIGINSTATE TEXT,
        ORIGINSTATENAME TEXT,
        DESTAIRPORTCODE TEXT,
        DESTAIRPORTNAME TEXT,
        DESTCITYNAME TEXT,
        DESTSTATE TEXT,
        DESTSTATENAME TEXT,
        CRSDEPTIME INTEGER,
        DEPTIME INTEGER,
        DEPDELAY INTEGER,
        TAXIOUT INTEGER,
        WHEELSOFF INTEGER,
        WHEELSON INTEGER,
        TAXIIN INTEGER,
        CRSARRTIME INTEGER,
        ARRTIME INTEGER,
        ARRDELAY INTEGER,
```

```
        CRSELAPSEDTIME INTEGER,
        ACTUALELAPSEDTIME INTEGER,
        CANCELLED TEXT,
        DIVERTED TEXT,
        DISTANCE TEXT
);

-- To delete all rows, restart the row_number
TRUNCATE FLIGHTS
RESTART IDENTITY;
```

Then, to import all data from flat file flights.txt

\copy FLIGHTS FROM 'D:\flights.txt' DELIMITER '|' CSV HEADER;

2. Create and load one fact table named FACT_FLIGHTS to contain data about the flights.
   See page 6

3. Create and load appropriate dimension table(s) named DIM_*.
   See page 7

4. Create a view named VW_FLIGHTS that joins your fact table to your dimension tables and returns columns useful for analysis. Please see the “View” section for more information.
   See page 9

Additional Details and Instructions

Fact Table

1. Create an additional column named DISTANCEGROUP that bins the distance values into groups in 100-mile increments. Example: 94 miles is 0-100 miles. 274 miles is 201-300 miles. Please make special note of the bins: 0-100, 201-300, 301-400, etc. Please make sure that the format of the bins is “201-300 miles”. Not “201-300”.

The fact table contains business facts (or measures)

-- Binning of miles to DISTANCEGROUP

```
SELECT DISTINCT(distance), CAST(SUBSTRING(DISTANCE FROM ('[0-9]+')) AS INT) ,
        CASE WHEN CAST(SUBSTRING(DISTANCE FROM ('[0-9]+')) AS INT) <= 100 THEN '0-100 miles'
        WHEN CAST(SUBSTRING(DISTANCE FROM ('[0-9]+')) AS INT) >= 10001 THEN '10000+ miles'
        WHEN LENGTH(SUBSTRING(DISTANCE FROM ('[0-9]+'))) = 3 THEN
        CASE
                WHEN SUBSTRING(SUBSTRING(DISTANCE FROM ('[0-9]+')), 2, 2) LIKE '00' THEN
                CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{1}')) AS INT) - 1) AS TEXT) || '01-' ||
                CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{1}')) AS INT)) AS TEXT) || '00 miles'
                ELSE
                SUBSTRING(DISTANCE FROM ('[0-9]{1}')) || '01-' ||
                CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{1}')) AS INT)+1) AS TEXT) || '00 miles'
        END

        WHEN SUBSTRING(SUBSTRING(DISTANCE FROM ('[0-9]+')), 2, 3) LIKE '_00' THEN
        CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{2}')) AS INT) - 1) AS TEXT) || '01-' ||
        CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{2}')) AS INT)) AS TEXT) || '00 miles'

        ELSE
        SUBSTRING(DISTANCE FROM ('[0-9]{2}')) || '01-' ||
        CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{2}')) AS INT)+1) AS TEXT) || '00 miles'

        END
        AS DISTANCEGROUP
FROM FLIGHTS
LIMIT 100;
```

2. Create an additional column named DEPDELAYGT15 that indicates (0/1) if the departure delay in minutes (DEPDELAY) is greater than 15.
   SELECT DEPDELAY, CASE WHEN DEPDELAY<=15 THEN 0 ELSE 1 END AS DEPDELAYGT15
   FROM FLIGHTS
   LIMIT 100;

3. Create an additional column named NEXTDAYARR that indicates (0/1) if the flight arrival time (ARRTIME) is the next day after the departure time (DEPTIME).

```
SELECT *, CASE WHEN DEPTIME-ARRTIME < 0
                    OR ARRTIME ISNULL OR DEPTIME ISNULL
                    THEN 0 ELSE 1 END AS NEXTDAYARR
FROM FLIGHTS;
```

4. Choose appropriate data types and perform conversions to load the data from the source into these types.

```
CREATE TABLE FLIGHTS
(       TRANSACTIONID INTEGER PRIMARY KEY,
        FLIGHTDATE DATE,
        AIRLINECODE TEXT,
        AIRLINENAME TEXT,
        TAILNUM TEXT,
        FLIGHTNUM INTEGER,
        ORIGINAIRPORTCODE TEXT,
        ORIGAIRPORTNAME TEXT,
        ORIGINCITYNAME TEXT,
        ORIGINSTATE TEXT,
        ORIGINSTATENAME TEXT,
        DESTAIRPORTCODE TEXT,
        DESTAIRPORTNAME TEXT,
        DESTCITYNAME TEXT,
        DESTSTATE TEXT,
        DESTSTATENAME TEXT,
        CRSDEPTIME INTEGER,
        DEPTIME INTEGER,
        DEPDELAY INTEGER,
        TAXIOUT INTEGER,
        WHEELSOFF INTEGER,
        WHEELSON INTEGER,
        TAXIIN INTEGER,
        CRSARRTIME INTEGER,
        ARRTIME INTEGER,
        ARRDELAY INTEGER,
        CRSELAPSEDTIME INTEGER,
        ACTUALELAPSEDTIME INTEGER,
        CANCELLED INTEGER,
        DIVERTED INTEGER,
        DISTANCE TEXT
);
```

5. Fixed bad data when encountered, such as:

1. Fixed TAILNUM, remove junk characters

```
SELECT TAILNUM, SUBSTRING(TAILNUM FROM '[a-zA-Z0-9]+') AS TAILNUM
FROM FLIGHTS
```

2. Make CANCELLED and DIVERTED streamlined with 'FALSE' or 'TRUE' values, making it boolean.

```
UPDATE FLIGHTS
SET CANCELLED = CASE WHEN LOWER(CANCELLED) IN ('f', '0', 'false') THEN FALSE ELSE TRUE END,
DIVERTED = CASE WHEN LOWER(DIVERTED) IN ('f', '0', 'false') THEN FALSE ELSE TRUE END;

ALTER TABLE FLIGHTS
ALTER COLUMN CANCELLED TYPE BOOLEAN USING CANCELLED:: BOOLEAN,
ALTER COLUMN DIVERTED TYPE BOOLEAN USING DIVERTED:: BOOLEAN;
```

4. Remove redundant AIRLINECODE from AIRLINENAME

```
SELECT AIRLINENAME, SUBSTRING(AIRLINENAME FROM '.*(?=\:)')
FROM FLIGHTS;
```

5. Remove redundant ORIGINCITYNAME and ORIGINSTATE from ORIGAIRPORTNAME

```
SELECT ORIGAIRPORTNAME, SUBSTRING(ORIGAIRPORTNAME FROM '(?<=\: ).*')
FROM FLIGHTS;
```

6. Remove redundant DESTCITYNAME and DESTSTATE from DESTAIRPORTNAME

```
SELECT DESTAIRPORTNAME, SUBSTRING(DESTAIRPORTNAME FROM '(?<=\: ).*')
FROM FLIGHTS;
```

7. COALESCE on TAILNUM column for null values, replaced with 'UNKOW'

```
COALESCE(SUBSTRING(TAILNUM FROM '[a-zA-Z0-9]+'), 'UNKNOW') AS TAILNUM
```

Dimension Table(s)

Dimension tables contain descriptive attributes (or fields) that are typically textual fields (or discrete numbers that behave like text)

1. Create at least one dimension table and load it from the source data.

```
CREATE TABLE DIM_AIRLINE
(       AIRLINECODE TEXT PRIMARY KEY,
        AIRLINENAME TEXT
);

CREATE TABLE DIM_AIRPORTS
```

```
(       AIRPORTCODE TEXT PRIMARY KEY,
        AIRPORTNAME TEXT,
        CITYNAME TEXT,
        STATE TEXT,
        STATENAME TEXT
);
```

2. Use your judgment about what columns from the source data should end up in the dimension tables. Be prepared to explain your decisions.

3. Clean up the AIRLINENAME column by removing the airline code from it.
   Explained above, using SUBSTRING(AIRLINENAME FROM '.*(?=\:)')

4. Clean up the ORIGAIRPORTNAME and DESTAIRPORTNAME columns by removing the concatenated city and state.
   Explained above, using SUBSTRING(DESTAIRPORTNAME FROM '(?<=\:).*')

5. Fix obviously bad data when encountered, if possible. Note these instances.

I've managed to make 2 FACT tables and 2 DIM tables with appropriate naming convention.
The commands to make and populate these table were as follows:

1. FACT_FLIGHTS

```
CREATE TABLE FACT_FLIGHTS
(       TRANSACTIONID INTEGER PRIMARY KEY,
        FLIGHTDATE DATE,
        AIRLINECODE TEXT,
        TAILNUM TEXT,
        FLIGHTNUM INTEGER,
        ORIGINAIRPORTCODE TEXT,
        DESTAIRPORTCODE TEXT,
        CANCELLED BOOL,
        DIVERTED BOOL
);

INSERT INTO FACT_FLIGHTS
SELECT
        TRANSACTIONID,
        FLIGHTDATE,
        AIRLINECODE,
        COALESCE(SUBSTRING(TAILNUM FROM '[a-zA-Z0-9]+'), 'UNKNOW') AS TAILNUM,
        FLIGHTNUM,
        ORIGINAIRPORTCODE,
        DESTAIRPORTCODE,
```

```
        CASE WHEN CANCELLED IN ('f', '0', 'false') THEN FALSE ELSE TRUE END AS CANCELLED,
        CASE WHEN DIVERTED IN ('f', '0', 'false') THEN FALSE ELSE TRUE END AS DIVERTED
FROM FLIGHTS
ORDER BY TRANSACTIONID;
```

2. FACT_FLIGHTRUNWAY

```
CREATE TABLE FACT_FLIGHTRUNWAY
(       TRANSACTIONID INTEGER PRIMARY KEY,
        CRSDEPTIME INTEGER,
        DEPTIME INTEGER,
        DEPDELAY INTEGER,
        TAXIOUT INTEGER,
        WHEELSOFF INTEGER,
        WHEELSON INTEGER,
        TAXIIN INTEGER,
        CRSARRTIME INTEGER,
        ARRTIME INTEGER,
        ARRDELAY INTEGER,
        CRSELAPSEDTIME INTEGER,
        ACTUALELAPSEDTIME INTEGER,
        DISTANCE TEXT,
        DISTANCEGROUP TEXT,
        DEPDELAYGT15 INTEGER,
        NEXTDAYARR INTEGER
);

INSERT INTO FACT_FLIGHTRUNWAY
SELECT
        TRANSACTIONID,CRSDEPTIME,DEPTIME,DEPDELAY,TAXIOUT,WHEELSOFF,WHEELSON,TAXIIN,
        CRSARRTIME,ARRTIME,ARRDELAY,CRSELAPSEDTIME,ACTUALELAPSEDTIME,DISTANCE,
        CASE WHEN CAST(SUBSTRING(DISTANCE FROM ('[0-9]+')) AS INT) <= 100 THEN '0-100 miles'
        WHEN CAST(SUBSTRING(DISTANCE FROM ('[0-9]+')) AS INT) >= 10001 THEN '10000+ miles'
        WHEN LENGTH(SUBSTRING(DISTANCE FROM ('[0-9]+'))) = 3 THEN
        CASE
                WHEN SUBSTRING(SUBSTRING(DISTANCE FROM ('[0-9]+')), 2, 2) LIKE '00' THEN
                CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{1}')) AS INT) - 1) AS TEXT) || '01-' ||
                CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{1}')) AS INT)) AS TEXT) || '00 miles'
                ELSE
                SUBSTRING(DISTANCE FROM ('[0-9]{1}')) || '01-' ||
                CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{1}')) AS INT)+1) AS TEXT) || '00 miles'
        END
        WHEN SUBSTRING(SUBSTRING(DISTANCE FROM ('[0-9]+')), 2, 3) LIKE '_00' THEN
        CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{2}')) AS INT) - 1) AS TEXT) || '01-' ||
```

```
        CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{2}')) AS INT)) AS TEXT) || '00 miles'
        ELSE
        SUBSTRING(DISTANCE FROM ('[0-9]{2}')) || '01-' ||
        CAST((CAST(SUBSTRING(DISTANCE FROM ('[0-9]{2}')) AS INT)+1) AS TEXT) || '00 miles'
        END
        AS DISTANCEGROUP,
        CASE WHEN DEPDELAY<=15 THEN 0 ELSE 1 END AS DEPDELAYGT15,
        CASE WHEN DEPTIME-ARRTIME < 0
                                OR ARRTIME ISNULL OR DEPTIME ISNULL
                                THEN 0 ELSE 1 END AS NEXTDAYARR
FROM FLIGHTS
ORDER BY TRANSACTIONID;
```

3. DIM_AIRLINES

```
CREATE TABLE DIM_AIRLINES
(       AIRLINECODE TEXT PRIMARY KEY,
        AIRLINENAME TEXT
);

INSERT INTO DIM_AIRLINES
SELECT DISTINCT
        AIRLINECODE,
        SUBSTRING(AIRLINENAME FROM '.*(?=\:)') AS AIRLINENAME
FROM FLIGHTS
ORDER BY AIRLINECODE;
```

4. DIM_AIRPORTS

```
CREATE TABLE DIM_AIRPORTS
(       AIRPORTCODE TEXT PRIMARY KEY,
        AIRPORTNAME TEXT,
        CITYNAME TEXT,
        STATE TEXT,
        STATENAME TEXT
);

INSERT INTO DIM_AIRPORTS
SELECT DISTINCT
        DESTAIRPORTCODE AS AIRPORTCODE,
        SUBSTRING(DESTAIRPORTNAME FROM '(?<=\: ).*') AS AIRPORTNAME,
        DESTCITYNAME AS CITYNAME,
        DESTSTATE AS STATE,
        DESTSTATENAME AS STATENAME
FROM FLIGHTS
ORDER BY DESTAIRPORTCODE;
```

View

My final view (VW_FLIGHTS) contains all the following columns:

- TRANSACTIONID
- DISTANCE
- DISTANCEGROUP
- DEPDELAY
- ARRDELAY
- DEPDELAYGT15
- NEXTDAYARR
- AIRLINENAME
- ORIGAIRPORTNAME
- DESTAIRPORTNAME
- ORIGINSTATE
- DESTSTATE
- CANCELLED
- DIVERTED

```
CREATE OR REPLACE VIEW VW_FLIGHTS AS
(       SELECT
        FF.TRANSACTIONID,
        (SELECT F.DISTANCE FROM FLIGHTS F WHERE F.TRANSACTIONID = FF.TRANSACTIONID) AS
        DISTANCE,
        FR.DISTANCEGROUP,
        FR.DEPDELAY,
        FR.ARRDELAY,
        FR.DEPDELAYGT15,
        FR.NEXTDAYARR,
        (SELECT AR.AIRLINENAME FROM DIM_AIRLINES AR WHERE FF.AIRLINECODE = AR.AIRLINECODE),
        (SELECT AP.AIRPORTNAME FROM DIM_AIRPORTS AP WHERE FF.ORIGINAIRPORTCODE =
        AP.AIRPORTCODE) AS ORIGAIRPORTNAME,
        (SELECT AP.AIRPORTNAME FROM DIM_AIRPORTS AP WHERE FF.DESTAIRPORTCODE =
        AP.AIRPORTCODE) AS DESTAIRPORTNAME,
        (SELECT AP.STATE FROM DIM_AIRPORTS AP WHERE FF.ORIGINAIRPORTCODE = AP.AIRPORTCODE)
        AS ORIGINSTATE,
        (SELECT AP.STATE FROM DIM_AIRPORTS AP WHERE FF.DESTAIRPORTCODE = AP.AIRPORTCODE)
        AS DESTSTATE,
        FF.CANCELLED,
        FF.DIVERTED
        FROM FACT_FLIGHTS FF
        JOIN FACT_FLIGHTRUNWAY FR
        ON FF.TRANSACTIONID = FR.TRANSACTIONID
);
SELECT * FROM VW_FLIGHTS LIMIT 100;
```