# Extract Transform and Load Tutorial – ETL Data Quality and cleaning

# ETL – 1

This tutorial works with the insurance data sets S\_0\_DataSet1 and S\_0\_DataSet2 to support the reports such as: No of claims per year per location.

Each dataset represents data from insurance companies, one operates across multiple states in America the other is just from Washington (S\_0\_DataSet1 and S\_0\_DataSet2)

# ETL 1 – overview

In this first tutorial we will implement the data dictionary design. There is a template for this Data Dictionary 1.

This tutorial:

1. Load the raw data (data sets S\_0\_DataSet1 and S\_0\_DataSet2) into Apex
2. Explore the datasets and consider the data quality issues in each data set.
3. Identify the columns in each dataset that are required to support the report: No of claims per year per location.
4. EXTRACT: Create a dataset that contains JUST the data you need to work out how many claims per year per location from both data sets (call this S1\_StageArea). You could have a number of these ‘stageing’ datasets)
5. Use code or tools, to find data quality issues with the data sets S1\_StageArea*. Document these in the Data Dictionary using the Data Dictionary 1 template.*
6. Use code to clean (quality checks) S1\_StageArea.
7. Consider the use of a table to log the changes made to the data.
8. Create the star schema model for the exercise in QSEE and create the DDL for the SS. It is good practice to prefix the star schema tables with SS.

# Suggested Solution –

# 1. ETL1\_Claims\_Merge\_data

-- Extract just the data we need for report no\_of\_claims per year per location

-- explore

DESC S0\_DATASET1

SELECT CUSTOMER, STATE, EFFECTIVE\_TO\_DATE, TOTAL\_CLAIM\_AMOUNT FROM S0\_DATASET1;

DESC S0\_DATASET2;

SELECT CUSTOMER, STATE, EFFECTIVE\_TO\_DATE, TOTAL\_CLAIM\_AMOUNT FROM S0\_DATASET2;

-- put all into one table

-- the data columns are related so we can stack the data on top of each other

-- sometimes we would put it alongside - so many column, or bring over separate datasets

DROP TABLE S1\_STAGEAREA;

CREATE TABLE S1\_STAGEAREA AS SELECT CUSTOMER, STATE, EFFECTIVE\_TO\_DATE, TOTAL\_CLAIM\_AMOUNT FROM S0\_DATASET1;

-- I'm going to add a column to show where the data has come from

ALTER TABLE S1\_STAGEAREA

ADD DATASOURCE VARCHAR2(5);

UPDATE S1\_STAGEAREA SET DATASOURCE = 'DS1';

INSERT INTO S1\_STAGEAREA (SELECT CUSTOMER, STATE, EFFECTIVE\_TO\_DATE, TOTAL\_CLAIM\_AMOUNT, 'DS2' FROM S0\_DATASET2);

# 2. ETL1\_Claims\_transforms

-- Data Quality checks and transformation on S1\_STAGEAREA

-- Quality Check example -

-- Update the data, maybe set it to ‘not known’, or maybe delete it?

-- This is the code we want to do - don't run it just yet

UPDATE S1\_STAGEAREA

SET state = 'not known'

WHERE

state IS NULL

OR state = ''

OR state = '-';

Consider the use of a table to log the changes made to the data,

Should we log the changes we have made? We have a data dictionary.

If so we could use a trigger and a ‘ETL\_log’ table.

Solution:

-- log ETL changes

DROP table etl\_log cascade constraints;

CREATE TABLE etl\_log

(issue\_id NUMBER(5) NOT NULL,

table\_name VARCHAR2(20),

data\_error\_code NUMBER(5),

issue\_desc VARCHAR2(50),

issue\_date DATE,

issue\_status VARCHAR2(20),

status\_update\_date DATE);

drop sequence EL\_SEQ;

create sequence EL\_SEQ

start with 1

increment by 1

maxvalue 10000

minvalue 1;

-- Now as part of the ETL - T for transformation, we will do data quality checks and log them in a data issues table

-- we have decided to update S1\_STAGEAREA directly.

DROP TRIGGER S1\_stagearea.trg\_quality\_chk ;

-- START of data transformations

CREATE or REPLACE trigger trg\_quality\_chk

before update on S1\_STAGEAREA

for each row

begin

INSERT INTO ETL\_log

(issue\_id, table\_name, data\_error\_code, issue\_desc, issue\_date, issue\_status, status\_update\_date)

VALUES

(EL\_SEQ.nextval, 'S1\_STAGEAREA', '0', 'Quality checks', SYSDATE, 'completed', SYSDATE);

end;

-- Quality Check

UPDATE S1\_STAGEAREA SET state = 'NOT KNOWN' WHERE state IS NULL;

-- Transformation (or Quality Check?)

UPDATE S1\_STAGEAREA SET state = 'Washington' WHERE state ='WA';

-- the dates are a mess

DELETE FROM S1\_STAGEAREA WHERE effective\_to\_date IS NULL;

-- we only want the years so (this is not entirely correct but will do for now)

-- Transform dates into just the year

DROP TABLE S2\_STAGEAREA;

CREATE TABLE S2\_STAGEAREA AS SELECT CUSTOMER, STATE, EFFECTIVE\_TO\_DATE, TOTAL\_CLAIM\_AMOUNT, DATASOURCE FROM S1\_STAGEAREA;

DELETE FROM S2\_STAGEAREA;

INSERT INTO S2\_STAGEAREA (SELECT CUSTOMER, STATE, '2011', TOTAL\_CLAIM\_AMOUNT, DATASOURCE FROM S1\_STAGEAREA WHERE effective\_to\_date LIKE '%11');

INSERT INTO S2\_STAGEAREA (SELECT CUSTOMER, STATE, '2012', TOTAL\_CLAIM\_AMOUNT, DATASOURCE FROM S1\_STAGEAREA WHERE effective\_to\_date LIKE '%12');

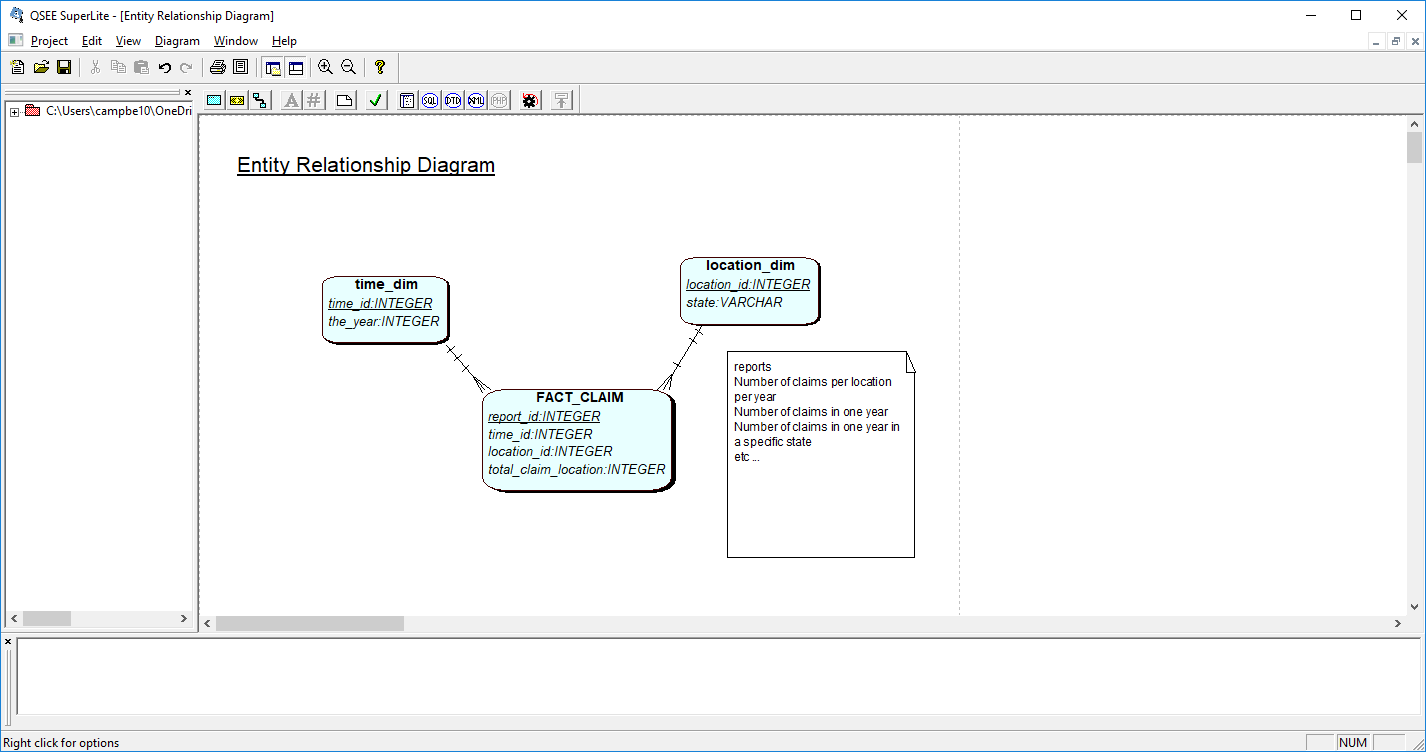
INSERT INTO S2\_STAGEAREA (SELECT CUSTOMER, STATE, '2010', TOTAL\_CLAIM\_AMOUNT, DATASOURCE FROM S1\_STAGEAREA WHERE effective\_to\_date LIKE '%10');

INSERT INTO S2\_STAGEAREA (SELECT CUSTOMER, STATE, '1960', TOTAL\_CLAIM\_AMOUNT, DATASOURCE FROM S1\_STAGEAREA WHERE effective\_to\_date LIKE '%60');

SELECT \* FROM S2\_STAGEAREA

SELECT \* FROM etl\_log;

# Create the star schema model for the exercise in QSEE and creates the DDL for the SS.



Use QSEE to create the above. Click on SQL button, change the generated script as you need to. (It needs changing!)

ETl 1 - Solution of DDL:

--------------------------------------------------------------

-- Database creation Script

-- Auto-Generated by QSEE-SuperLite (c) 2001-2004 QSEE-Technologies Ltd.

-- Verbose generation: ON

-- note: spaces within table/column names have been replaced by underscores (\_)

-- Target DB: SQL2

-- Entity Model :Entity Relationship Diagram

--------------------------------------------------------------

DROP table time\_dim cascade constraints;

DROP table location\_dim cascade constraints;

DROP table fact\_claim cascade constraints;

--------------------------------------------------------------

-- Table Creation --

-- Each entity on the model is represented by a table that needs to be created within the Database.

-- Within SQL new tables are created using the CREATE TABLE command.

-- When a table is created its name and its attributes are defined.

-- The values of which are derived from those specified on the model.

-- Certain constraints are sometimes also specified, such as identification of primary keys.

-- Create a Database table to represent the "FACT\_CLAIM" entity.

CREATE TABLE FACT\_CLAIM(

report\_id INTEGER NOT NULL,

fk1\_time\_id INTEGER NOT NULL,

fk2\_location\_id INTEGER NOT NULL,

total\_claim\_location INTEGER,

-- Specify the PRIMARY KEY constraint for table "FACT\_CLAIM".

-- This indicates which attribute(s) uniquely identify each row of data.

CONSTRAINT pk\_FACT\_CLAIM PRIMARY KEY (report\_id)

);

-- Create a Database table to represent the "time\_dim" entity.

CREATE TABLE time\_dim(

time\_id INTEGER NOT NULL,

the\_year INTEGER,

-- Specify the PRIMARY KEY constraint for table "time\_dim".

-- This indicates which attribute(s) uniquely identify each row of data.

CONSTRAINT pk\_time\_dim PRIMARY KEY (time\_id)

);

-- Create a Database table to represent the "location\_dim" entity.

CREATE TABLE location\_dim(

location\_id INTEGER NOT NULL,

state VARCHAR(10),

-- Specify the PRIMARY KEY constraint for table "location\_dim".

-- This indicates which attribute(s) uniquely identify each row of data.

CONSTRAINT pk\_location\_dim PRIMARY KEY (location\_id)

);

--------------------------------------------------------------

-- Alter Tables to add fk constraints –

# 3. ETL1\_Claims\_load

-- Populate the Dimension tables

-- Populate the fact table

-- first run the SS claims script to create the dimension and fact table

-- Now to populate the time\_dim

DROP sequence time\_seq;

create sequence time\_SEQ

start with 1

increment by 1

maxvalue 10000

minvalue 1;

INSERT INTO time\_dim VALUES( time\_seq.nextval, '2011');

INSERT INTO time\_dim VALUES( time\_seq.nextval, '2010');

INSERT INTO time\_dim VALUES( time\_seq.nextval, '2012');

INSERT INTO time\_dim VALUES( time\_seq.nextval, '1960');

-- Now repeat similar for the location dimension table

--

DROP table tmp\_locations;

Create table tmp\_locations as SELECT DISTINCT state FROM S2\_STAGEAREA;

DROP sequence loc\_seq;

create sequence loc\_SEQ

start with 1

increment by 1

maxvalue 10000

minvalue 1;

INSERT INTO location\_dim SELECT loc\_seq.nextval, state FROM tmp\_locations;

-- Or use straight forward code as below

-- INSERT INTO location\_dim VALUES (1, 'WA');

-- Populates the Fact table from the cleaned data sets

-- the report in sum of the claims per year

-- Coding approaches are discussed and evaluated

-- Solutions:

-- Option1: Create a detail table, then select from this.

-- Option2: Create a cart-prod of ids from time and location and then update on the fact table.

-- Option1 – solution as below

-- The fact\_claims table

DROP sequence fact\_seq;

create sequence FACT\_SEQ

start with 1

increment by 1

maxvalue 10000

minvalue 1;

DROP table tmp\_claim1;

-- this tmp table contains all data we need, with the year (not the whole date)

CREATE TABLE tmp\_claim1 AS SELECT state, total\_claim\_amount, effective\_to\_date as which\_year

FROM S2\_STAGEAREA;

DROP table tmp\_claim2;

-- This tmp table contains the sum of the claim by year and state

CREATE TABLE tmp\_claim2 AS

SELECT which\_year, state, SUM(total\_claim\_amount) as claim\_amount FROM tmp\_claim1

GROUP BY which\_year, state;

-- check the data

SELECT \* FROM tmp\_claim2;

SELECT \* FROM time\_dim;

SELECT \* FROM location\_dim;

INSERT INTO FACT\_claim (report\_id, fk1\_time\_id, fk2\_location\_id , total\_claim\_location)

SELECT fact\_seq.nextval, time\_dim.time\_id, location\_dim.location\_id, tmp\_claim2.claim\_amount

FROM tmp\_claim2, time\_dim, location\_dim

WHERE tmp\_claim2.which\_year = time\_dim.the\_year AND tmp\_claim2.state = location\_dim.state;

-- Over to you to check run as reports

SELECT \* FROM fact\_claim

Lastly consider how data can be extracted from the SS tables and pulled into a visualisation tool, as we did with Excel.