# CASE STUDY: Pizza Runner

**DATA CLEANING WITH SQL**

Sege is the owner of a pizza business. He knew that just selling pizza was not going to help him get seed funding to expand his new Pizza Empire - so he had one more genius idea to combine with it - he was going to Uberize it - and so Pizza Runner was launched!. Danny started by recruiting “runners” to deliver fresh pizza from Pizza Runner Headquarters he paid for the building of a mobile app to accept orders from customers.

Danny has collected data for his business but he requires further assistance to clean his data.

There are six tables in total: runners, customer\_orders, runner\_orders, pizza\_names, pizza\_recipes, pizza\_toppings.

**Pre-Cleaning Process**

First, all tables were accessed to identify tables that needed data cleaning. Three tables needed data cleaning

**Issues Identified:**

Customer\_orders: blanks and null values, faulty data type, double information in exclusions and extras column, split datetime column to date and time only. Also, no primary key was observed.

Runner\_orders: blanks and null values, split pickup\_time datetime column, normalize distance and duration column, remove km and mins from both, then convert to numbers from string.

Pizza\_recipe: split topping column. One row and column must hold just and distinct data information. Convert table to long

**DATA CLEANING**

**--Create new table which holds a copy of the customer\_orders table. This is to ensure preservation of the original dataset.**

DROP TABLE IF EXISTS clean\_customer\_orders;

SELECT \*

INTO clean\_customer\_orders

FROM pizza\_runner.customer\_orders

**--confirm correct creation of table copy**

SELECT \*

FROM clean\_customer\_orders

**-- Create new column "date\_ordered" and fill same with values from the split order\_time column**

ALTER TABLE clean\_customer\_orders

ADD date\_ordered Date

UPDATE clean\_customer\_orders

SET date\_ordered = cast(order\_time as Date)

**-- Create new column "time\_ordered" and fill same with values from the split order\_time column**

ALTER TABLE clean\_customer\_orders

ADD time\_ordered Time

UPDATE clean\_customer\_orders

SET time\_ordered = cast(order\_time as time)

**--Confirm successful addition of both columns**

SELECT \*

FROM clean\_customer\_orders

**--Update nulls and blanks to NULL in exclusions column of the clean\_customer\_orders table**

UPDATE clean\_customer\_orders

SET exclusions = CASE when exclusions = ''

OR exclusions = 'null' THEN NULL

ELSE exclusions

END::varchar(23);

**--Update nulls and blanks to NULL in extras column in the clean\_customer\_orders table**

UPDATE clean\_customer\_orders

SET extras = CASE when extras = ''

OR extras = 'null' THEN NULL

ELSE extras

END::varchar(23);

**-- Create new column "exclusions\_count " and fill same with values that count the number of exclusions per order in the clean\_customer\_orders table.**

ALTER TABLE clean\_customer\_orders

ADD exclusions\_count integer

**-- Fill new columns with the exclusion count**

UPDATE clean\_customer\_orders

SET exclusions\_count = CAST(LENGTH(REPLACE((REPLACE(exclusions, ' ', '')),',','')) as integer)

**-- Create new column "extras\_count " and fill same with values that count the number of extras per order in the clean\_customer\_orders table.**

ALTER TABLE clean\_customer\_orders

ADD extras\_count integer

**-- Fill new columns with the extras count**

UPDATE clean\_customer\_orders

SET extras\_count = CAST(LENGTH(REPLACE((REPLACE(extras, ' ', '')),',','')) as integer)

**--Create table “multiple\_exclusions\_extras”, holding orders with exclusions or extras greater than 1 to address the denormalization in the clean\_customer\_orders table and unnest cells that hold multiple comma separated values.**

DROP TABLE IF EXISTS multiple\_exclusions\_extras;

SELECT order\_id,

UNNEST(STRING\_TO\_ARRAY(exclusions, ',')) AS exclusions,

UNNEST(STRING\_TO\_ARRAY(extras, ',')) AS extras

INTO multiple\_exclusions\_extras

FROM clean\_customer\_orders

WHERE exclusions\_count > 1 or extras\_count > 1;

**--Confirm successful creation of table**

SELECT \*

FROM multiple\_exclusions\_extras

**--Create new table which holds a copy of the pizza\_recipes table. This is to ensure preservation of the original dataset.**

DROP TABLE IF EXISTS clean\_ pizza\_recipes;

SELECT \*

INTO clean\_ pizza\_recipes

FROM pizza\_runner.pizza\_recipes

**--confirm correct creation of table copy**

SELECT \*

FROM clean\_ pizza\_recipes

**--unnest pizza\_id and toppings column in clean\_ pizza\_recipes table**

SELECT pizza\_id,

UNNEST(STRING\_TO\_ARRAY(TOPPINGS, ',')) AS TOPPINGS

FROM clean\_ pizza\_recipes

**--Create new table which holds a copy of the runner\_orders table. This is to ensure preservation of the original dataset.**

DROP TABLE IF EXISTS clean\_runner\_orders

SELECT \*

INTO clean\_runner\_orders

FROM pizza\_runner.runner\_orders

**--confirm correct creation of table copy**

SELECT \*

FROM clean\_runner\_orders

**--Update nulls and blanks to NULL in pickup\_time column of the clean\_runner\_orders table**

UPDATE clean\_runner\_orders

SET pickup\_time = case when pickup\_time = ''

or pickup\_time = 'null' then NULL

else pickup\_time

END::timestamp ;

**--Update nulls and blanks to NULL in distance column and strip off km in the clean\_runner\_orders table**

UPDATE clean\_runner\_orders

SET distance = CASE WHEN distance = '' OR distance = 'null' THEN NULL

WHEN distance LIKE '%km' THEN TRIM('km' from distance)

ELSE distance

END :: float

**--Update nulls and blanks to NULL in cancellation column of the clean\_runner\_orders table**

UPDATE clean\_runner\_orders

SET cancellation = CASE WHEN cancellation = '' OR cancellation = 'null' THEN NULL

ELSE cancellation

END :: varchar(23)

**--Update nulls and blanks to NULL in duration column of the clean\_runner\_orders table**

UPDATE clean\_runner\_orders

SET duration = CASE WHEN duration = '' OR duration = 'null' THEN NULL

ELSE duration

END :: varchar(23)

**--strip duration column of non numeric characters in the clean\_runner\_orders table**

UPDATE clean\_runner\_orders

SET duration = CASE WHEN duration <> '' THEN LEFT(duration, 2)

ELSE duration

END :: integer;

**--Alter the data types of pickup\_time, distance and duration columns of the clean\_runner\_orders table**

ALTER TABLE clean\_runner\_orders

ALTER COLUMN pickup\_time TYPE TIMESTAMP USING pickup\_time:: timestamp,

ALTER COLUMN distance TYPE FLOAT USING distance:: double precision,

ALTER COLUMN duration TYPE INT USING duration:: integer;

**-- Create new column "date\_pickup" and fill same with values from the split pickup\_time column**

ALTER TABLE clean\_runner\_orders

ADD date\_pickup Date

UPDATE clean\_runner\_orders

SET date\_pickup = cast(pickup\_time as Date)

**-- Create new column "time\_pickup" and fill same with values from the split pickup\_time column**

ALTER TABLE clean\_runner\_orders

ADD time\_pickup Time

UPDATE clean\_runner\_orders

SET time\_pickup = CAST(pickup\_time as time)

**--Confirm successful addition of both columns**

SELECT \*

FROM clean\_runner\_orders

**CONCLUSION**

After following laid down processes for data cleaning using sql, the datasets are now good enough to be used to generate insights and drive data-driven decision making.

These datasets were gotten from the Danny Ma’s 8 weeks SQL challenge.

The Schema SQL code for the creation of permanent tables used for this project is attached below

**CREATE** **SCHEMA** pizza\_runner;

**SET** search\_path = pizza\_runner;

**DROP** **TABLE** **IF** **EXISTS** runners;

**CREATE** **TABLE** runners (

"runner\_id" INTEGER,

"registration\_date" DATE

);

**INSERT** **INTO** runners

("runner\_id", "registration\_date")

**VALUES**

(1, '2021-01-01'),

(2, '2021-01-03'),

(3, '2021-01-08'),

(4, '2021-01-15');

**DROP** **TABLE** **IF** **EXISTS** customer\_orders;

**CREATE** **TABLE** customer\_orders (

"order\_id" INTEGER,

"customer\_id" INTEGER,

"pizza\_id" INTEGER,

"exclusions" VARCHAR(4),

"extras" VARCHAR(4),

"order\_time" **TIMESTAMP**

);

**INSERT** **INTO** customer\_orders

("order\_id", "customer\_id", "pizza\_id", "exclusions", "extras", "order\_time")

**VALUES**

('1', '101', '1', '', '', '2020-01-01 18:05:02'),

('2', '101', '1', '', '', '2020-01-01 19:00:52'),

('3', '102', '1', '', '', '2020-01-02 23:51:23'),

('3', '102', '2', '', NULL, '2020-01-02 23:51:23'),

('4', '103', '1', '4', '', '2020-01-04 13:23:46'),

('4', '103', '1', '4', '', '2020-01-04 13:23:46'),

('4', '103', '2', '4', '', '2020-01-04 13:23:46'),

('5', '104', '1', 'null', '1', '2020-01-08 21:00:29'),

('6', '101', '2', 'null', 'null', '2020-01-08 21:03:13'),

('7', '105', '2', 'null', '1', '2020-01-08 21:20:29'),

('8', '102', '1', 'null', 'null', '2020-01-09 23:54:33'),

('9', '103', '1', '4', '1, 5', '2020-01-10 11:22:59'),

('10', '104', '1', 'null', 'null', '2020-01-11 18:34:49'),

('10', '104', '1', '2, 6', '1, 4', '2020-01-11 18:34:49');

**DROP** **TABLE** **IF** **EXISTS** runner\_orders;

**CREATE** **TABLE** runner\_orders (

"order\_id" INTEGER,

"runner\_id" INTEGER,

"pickup\_time" VARCHAR(19),

"distance" VARCHAR(7),

"duration" VARCHAR(10),

"cancellation" VARCHAR(23)

);

**INSERT** **INTO** runner\_orders

("order\_id", "runner\_id", "pickup\_time", "distance", "duration", "cancellation")

**VALUES**

('1', '1', '2020-01-01 18:15:34', '20km', '32 minutes', ''),

('2', '1', '2020-01-01 19:10:54', '20km', '27 minutes', ''),

('3', '1', '2020-01-03 00:12:37', '13.4km', '20 mins', NULL),

('4', '2', '2020-01-04 13:53:03', '23.4', '40', NULL),

('5', '3', '2020-01-08 21:10:57', '10', '15', NULL),

('6', '3', 'null', 'null', 'null', 'Restaurant Cancellation'),

('7', '2', '2020-01-08 21:30:45', '25km', '25mins', 'null'),

('8', '2', '2020-01-10 00:15:02', '23.4 km', '15 minute', 'null'),

('9', '2', 'null', 'null', 'null', 'Customer Cancellation'),

('10', '1', '2020-01-11 18:50:20', '10km', '10minutes', 'null');

**DROP** **TABLE** **IF** **EXISTS** pizza\_names;

**CREATE** **TABLE** pizza\_names (

"pizza\_id" INTEGER,

"pizza\_name" TEXT

);

**INSERT** **INTO** pizza\_names

("pizza\_id", "pizza\_name")

**VALUES**

(1, 'Meatlovers'),

(2, 'Vegetarian');

**DROP** **TABLE** **IF** **EXISTS** pizza\_recipes;

**CREATE** **TABLE** pizza\_recipes (

"pizza\_id" INTEGER,

"toppings" TEXT

);

**INSERT** **INTO** pizza\_recipes

("pizza\_id", "toppings")

**VALUES**

(1, '1, 2, 3, 4, 5, 6, 8, 10'),

(2, '4, 6, 7, 9, 11, 12');

**DROP** **TABLE** **IF** **EXISTS** pizza\_toppings;

**CREATE** **TABLE** pizza\_toppings (

"topping\_id" INTEGER,

"topping\_name" TEXT

);

**INSERT** **INTO** pizza\_toppings

("topping\_id", "topping\_name")

**VALUES**

(1, 'Bacon'),

(2, 'BBQ Sauce'),

(3, 'Beef'),

(4, 'Cheese'),

(5, 'Chicken'),

(6, 'Mushrooms'),

(7, 'Onions'),

(8, 'Pepperoni'),

(9, 'Peppers'),

(10, 'Salami'),

(11, 'Tomatoes'),

(12, 'Tomato Sauce');