WIE3008

BUSINESS ANALYTICS AND INTELLIGENCE

ASSIGNMENT 2

Name: Amir Firdaus Bin Abdul Hadi

Matric Number: 17204620/2

Group Members

Imran Asyraaf Bin Mohd Ikhalan

Aiman Wafiq Bin Appandi

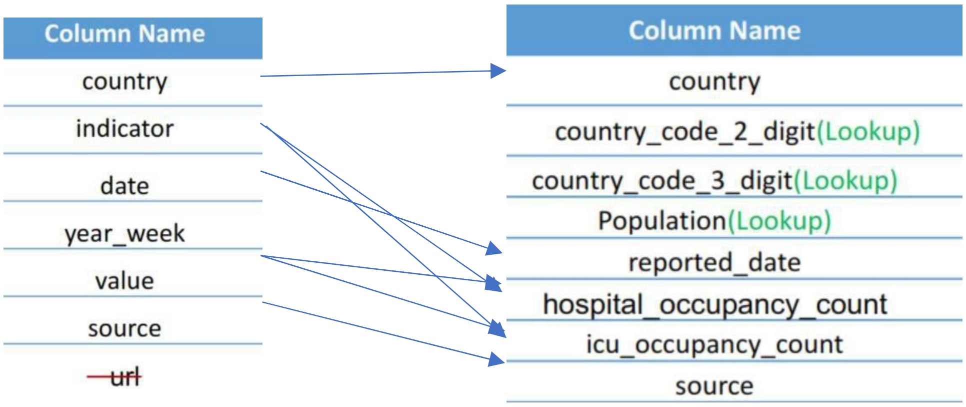
Irfan Abidin As Salik Bin Noor Riza Al Jeffery

Mohamad Nizar Mustaqeem Bin Mazlan

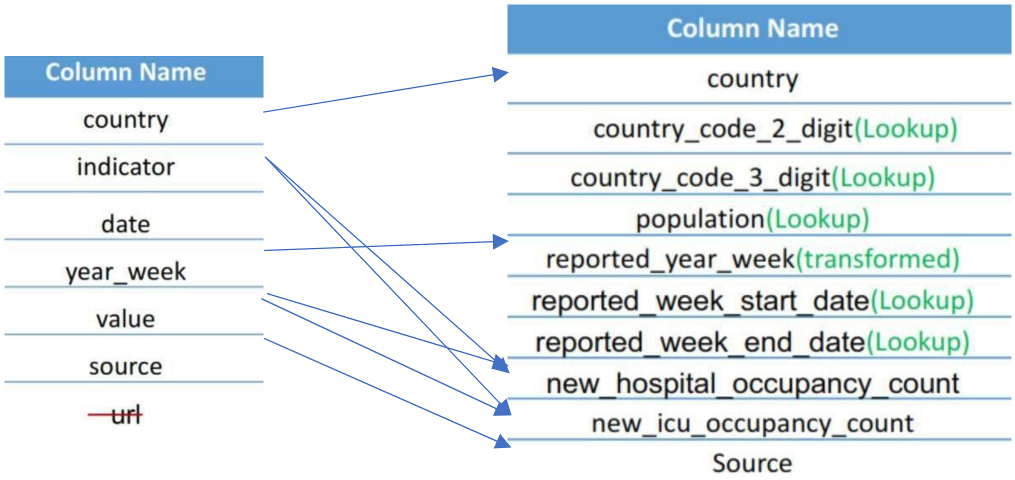
Part A (5 marks)

Building the Azure Data Factory pipeline.

1. Complete the pipeline by preparing the following sink datasets based on provided mapping and transformation requirements.
    a. Transformed Daily File



    b. Transformed Weekly File


Column Name
country
indicator
date
year_week
value
source
~~url~~
Column Name
country
country_code_2_digit(Lookup)
country_code_3_digit(Lookup)
population(Lookup)
reported_year_week(transformed)
reported_week_start_date(Lookup)
reported_week_end_date(Lookup)
new_hospital_occupancy_count
new_icu_occupancy_count
Source

a. Pipeline for Transformed Daily File:

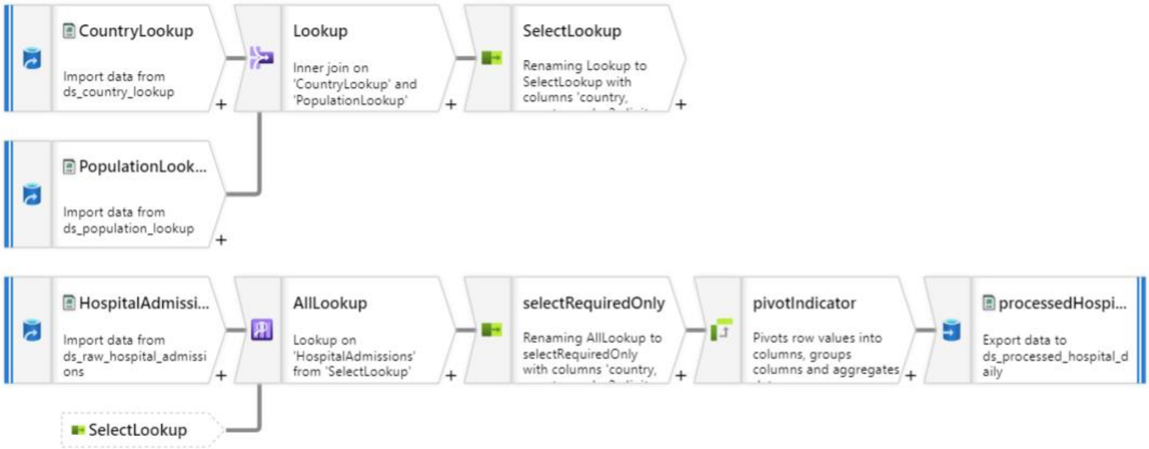

CountryLookup
Import data from ds_country_lookup
Lookup
Inner join on 'CountryLookup' and 'PopulationLookup'
SelectLookup
Renaming Lookup to SelectLookup with columns 'country,
PopulationLook...
Import data from ds_population_lookup
HospitalAdmissi...
Import data from ds_raw_hospital_admissions
AllLookup
Lookup on 'HospitalAdmissions' from 'SelectLookup'
selectRequiredOnly
Renaming AllLookup to selectRequiredOnly with columns 'country,
pivotIndicator
Pivots row values into columns, groups columns and aggregates
processedHospi...
Export data to ds_processed_hospital_daily
SelectLookup

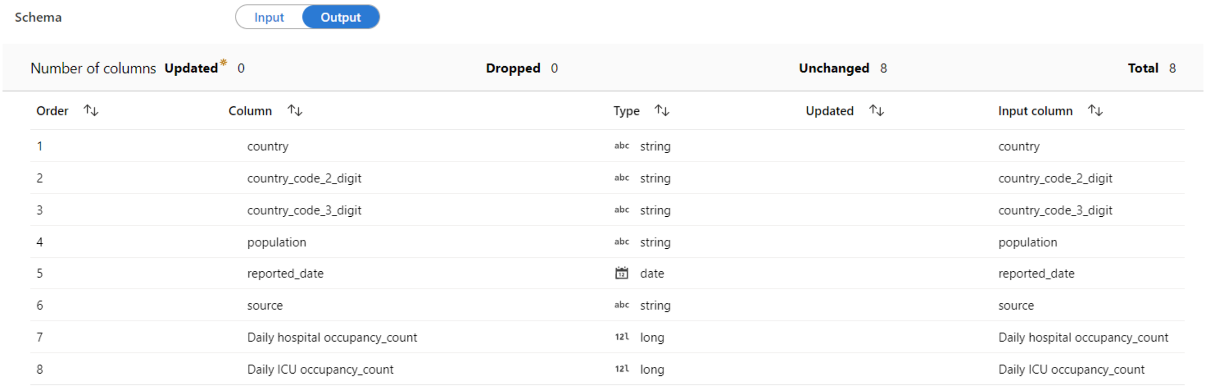
Figure 1: Pipeline for Hospital Admission Daily File

For this pipeline, we used 3 datasets to achieve the required sink result which are Country Lookup, Population Lookup and Hospital Admission. There are 3 lookup columns needed which are “country_code_2_digit”, “country_code_3_digit” and “population”. Country Lookup provides the lookup for “country_code_2_digit” and “country_code_3_digit” while Population Lookup provides the lookup for “population”. Hospital Admission dataset is obtained through http file and then stored into data lake.

Hospital Admission Http link:
https://raw.githubusercontent.com/cloudboxacademy/covid19/main/ecdc_data/hospital_admissions.csv

All Lookup file can be found in this drive:
https://drive.google.com/drive/folders/1nuXV8XIHCw6jrDM496jTmmCf8BkyMt6I?usp=share_link

First, we joined all lookup columns needed from Country Lookup and Population Lookup. Then, we select only required field after the lookup joining to remove any redundant and unnecessary columns. After that, to create “hospital_occupancy_count” and “icu_occupancy_count”, we pivoted the “indicator” column into 2 new columns based on the “value” column. Lastly, the processed data is sink as a transformed file. The final transformation output can be seen on figure 2 while the sample exported excel sink file on figure 3.

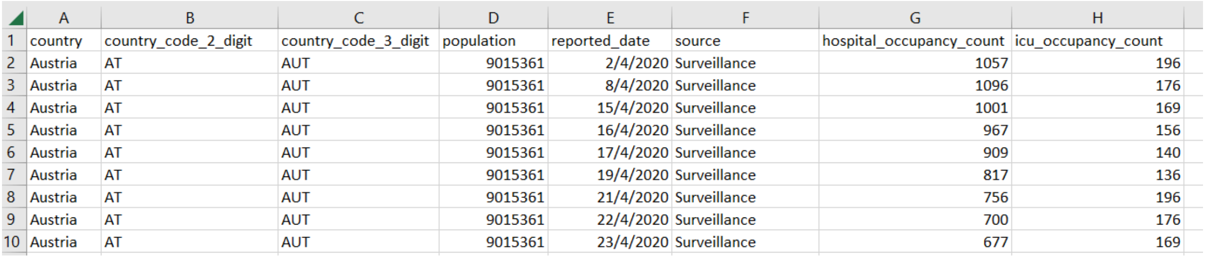


Schema Input Output

Number of columns Updated* 0 Dropped 0 Unchanged 8 Total 8

Order ↑↓	Column ↑↓	Type ↑↓	Updated ↑↓	Input column ↑↓
1	country	abc string		country
2	country_code_2_digit	abc string		country_code_2_digit
3	country_code_3_digit	abc string		country_code_3_digit
4	population	abc string		population
5	reported_date	date		reported_date
6	source	abc string		source
7	Daily hospital occupancy_count	12L long		Daily hospital occupancy_count
8	Daily ICU occupancy_count	12L long		Daily ICU occupancy_count

Figure 2: Final Transformation Output for Daily File



	A	B	C	D	E	F	G	H
1	country	country_code_2_digit	country_code_3_digit	population	reported_date	source	hospital_occupancy_count	icu_occupancy_count
2	Austria	AT	AUT	9015361	2/4/2020	Surveillance	1057	196
3	Austria	AT	AUT	9015361	8/4/2020	Surveillance	1096	176
4	Austria	AT	AUT	9015361	15/4/2020	Surveillance	1001	169
5	Austria	AT	AUT	9015361	16/4/2020	Surveillance	967	156
6	Austria	AT	AUT	9015361	17/4/2020	Surveillance	909	140
7	Austria	AT	AUT	9015361	19/4/2020	Surveillance	817	136
8	Austria	AT	AUT	9015361	21/4/2020	Surveillance	756	196
9	Austria	AT	AUT	9015361	22/4/2020	Surveillance	700	176
10	Austria	AT	AUT	9015361	23/4/2020	Surveillance	677	169

Figure 3: Sample of Exported Sink Daily File

b. Pipeline for Transformed Weekly File:

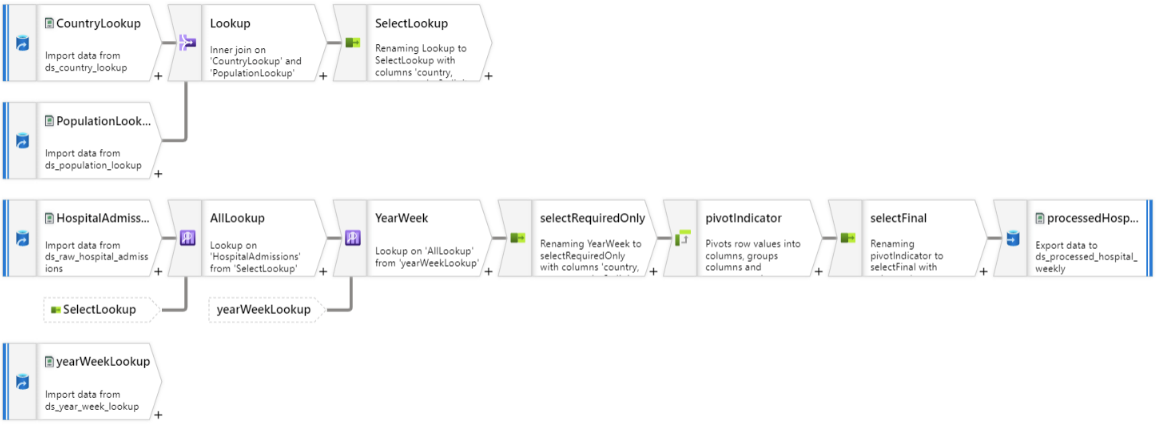
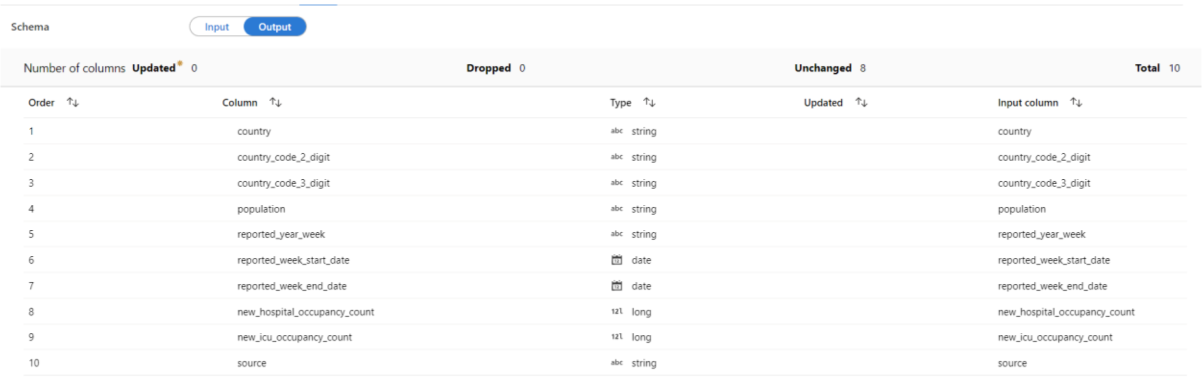

CountryLookup
Import data from ds_country_lookup
Lookup
Inner join on 'CountryLookup' and 'PopulationLookup'
SelectLookup
Renaming Lookup to SelectLookup with columns 'country,
PopulationLook...
Import data from ds_population_lookup
HospitalAdmiss...
Import data from ds_raw_hospital_admissions
AllLookup
Lookup on 'HospitalAdmissions' from 'SelectLookup'
YearWeek
Lookup on 'AllLookup' from 'yearWeekLookup'
selectRequiredOnly
Renaming YearWeek to selectRequiredOnly with columns 'country,
pivotIndicator
Pivots row values into columns, groups columns and
selectFinal
Renaming pivotIndicator to selectFinal with
processedHosp...
Export data to ds_processed_hospital_weekly
SelectLookup
yearWeekLookup
yearWeekLookup
Import data from ds_year_week_lookup

Figure 4: Pipeline for Hospital Admission Weekly File

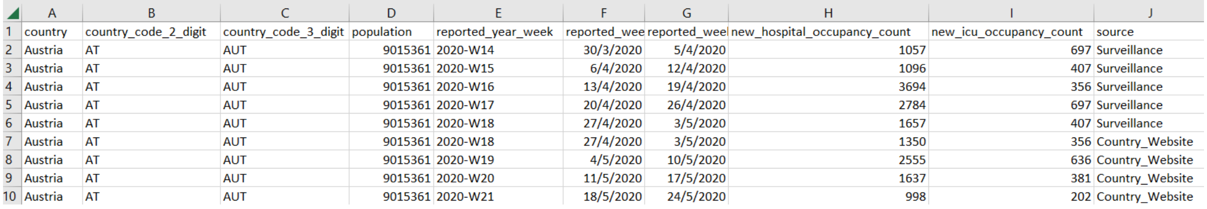
The process for this weekly pipeline is similar with daily pipeline with removal of reported date and addition of reported year week and its start and end date. For the week date, we add new dataset to the data lake called “yearWeekLookup” to provide the dates for the start and end date of the week. The rest is the same and the processed data is sink as a transformed weekly file. The final transformation output can be seen on figure 5 while the sample exported excel sink file on figure 6.



Number of columns Updated* 0 Dropped 0 Unchanged 8 Total 10

Order	Column	Type	Updated	Input column
1	country	abc string		country
2	country_code_2_digit	abc string		country_code_2_digit
3	country_code_3_digit	abc string		country_code_3_digit
4	population	abc string		population
5	reported_year_week	abc string		reported_year_week
6	reported_week_start_date	date		reported_week_start_date
7	reported_week_end_date	date		reported_week_end_date
8	new_hospital_occupancy_count	12l long		new_hospital_occupancy_count
9	new_icu_occupancy_count	12l long		new_icu_occupancy_count
10	source	abc string		source

Figure 5: Final Transformation Output for Weekly File



	A	B	C	D	E	F	G	H	I	J
1	country	country_code_2_digit	country_code_3_digit	population	reported_year_week	reported_wee	reported_weel	new_hospital_occupancy_count	new_icu_occupancy_count	source
2	Austria	AT	AUT	9015361	2020-W14	30/3/2020	5/4/2020	1057	697	Surveillance
3	Austria	AT	AUT	9015361	2020-W15	6/4/2020	12/4/2020	1096	407	Surveillance
4	Austria	AT	AUT	9015361	2020-W16	13/4/2020	19/4/2020	3694	356	Surveillance
5	Austria	AT	AUT	9015361	2020-W17	20/4/2020	26/4/2020	2784	697	Surveillance
6	Austria	AT	AUT	9015361	2020-W18	27/4/2020	3/5/2020	1657	407	Surveillance
7	Austria	AT	AUT	9015361	2020-W18	27/4/2020	3/5/2020	1350	356	Country_Website
8	Austria	AT	AUT	9015361	2020-W19	4/5/2020	10/5/2020	2555	636	Country_Website
9	Austria	AT	AUT	9015361	2020-W20	11/5/2020	17/5/2020	1637	381	Country_Website
10	Austria	AT	AUT	9015361	2020-W21	18/5/2020	24/5/2020	998	202	Country_Website

Figure 6: Sample of Exported Sink Weekly File

Part B (20 marks)

Build a new project to ingest e-commerce and currency exchange rate data for localized reporting. Source or dataset:

a. E-Commerce Business Transaction from Kaggle (https://www.kaggle.com/datasets/gabrielramos87/an-online-shop-business)
b. Exchange Rates by Bank Negara Malaysia (https://apikijangportal.bnm.gov.my/openapi?category=Rates%20and%20Volumes)
c. Country and Currency Code (to find relevant dataset from a trusted source)

High Level Requirements

1. To ingest both E-Commerce Business Transaction and Exchange Rates dataset into data lake (5 marks)
2. Map Country field from transaction to respective currency code and retrieve equivalent price in MYR according to date of exchange rate and transaction. (note: assume Price shown is in local currency of purchaser) (10 marks)
3. Get total price for each transaction in local currency and MYR (5 marks)

Expected sink datasets structure:

Daily Transaction

Field	Data Type	Description
TransactionNo	Integer	Unique number that defines each transaction
Date	Date	Transaction date
ProductNo	String	Unique number used to identify product
ProductName	String	Product name
PriceLocal	Decimal	Price in local currency of purchaser's country
CurrencyLocal	Decimal	Local 3-digit currency code of purchaser's country
PriceConverted	Decimal	MYR equivalent of transacted price
CurrencyConverted	Decimal	Default to 'MYR'
Quantity	Integer	Quantity of each product per transaction
ExchangeRate	Decimal	Currency exchange rate for transacted currency and date
TotalPriceLocal	Decimal	Total price = priceLocal * Quantity
TotalPriceConverted	Decimal	MYR equivalent of TotalPriceLocal
CustomerNo	Integer	Unique number to identify customer
Country	String	Country where customer reside

Weekly Summary

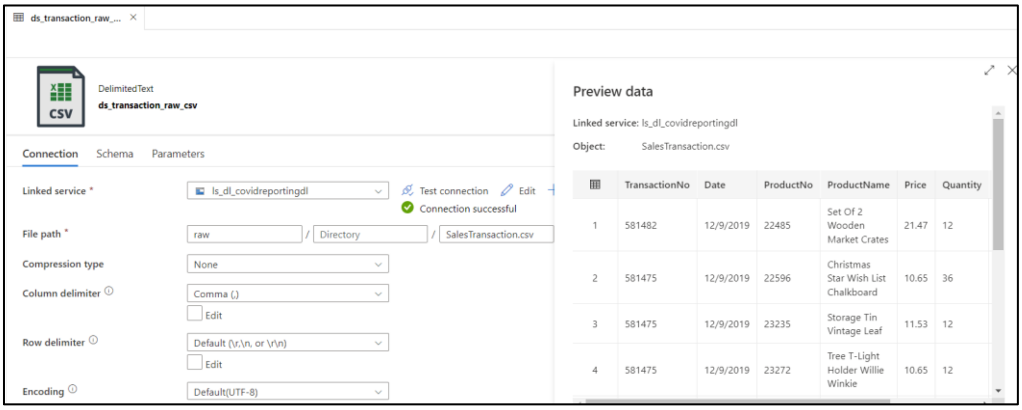
Sum of transaction by product and country.

Field	Data Type	Description
ProductNo	String	Unique number used to identify product
ProductName	String	Product name
PriceLocal	Decimal	Price in local currency of purchaser's country
TotalPriceLocal	Decimal	Total price = priceLocal * Quantity
TotalPriceConverted	Decimal	MYR equivalent of TotalPriceLocal
Country	String	Country where customer reside
ReportedMonth	Integer	Month of transaction
ReportedYear	Integer	Year of transaction

Daily Transaction:

Ingestion

For this project, we are going to need 3 datasets; business transaction from Kaggle, exchange rates from API link and currency code lookup that can be found online. Figure 1 shows how business transaction data ingested into data lake link service that has been created in Part A. For exchange rates, the API link given from BNM do not work as intended, so we are using other exchange rates API (https://api.exchangerate.host/latest). To ingest API dataset, we created a REST link service and provides the base URL of the API (Figure 2). Then, we created a pipeline (Figure 3) to copy data using the link service into data lake in the form of JSON file. Figure 4 display the preview of the JSON file that have been successfully ingested.



Linked service * ls_dl_covidreportingdl

Connection successful

File path * raw / Directory / SalesTransaction.csv

Compression type: None

Column delimiter: Comma (,) □ Edit

Row delimiter: Default (\r,\n, or \r\n) □ Edit

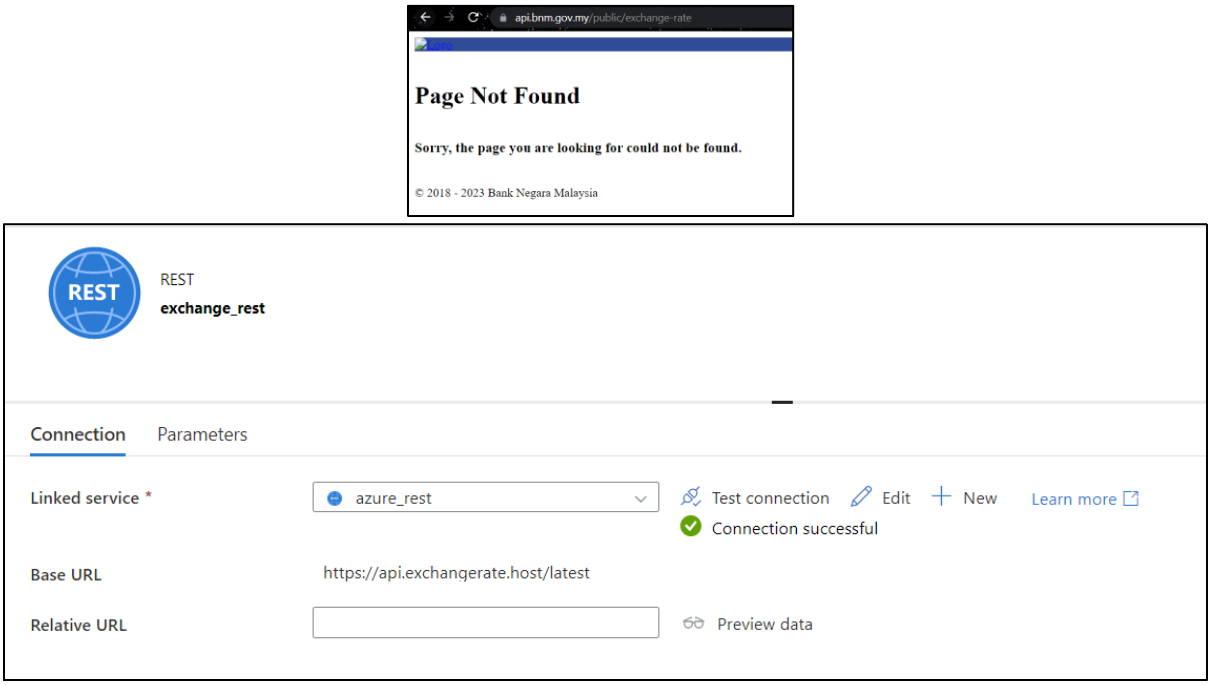
Encoding: Default(UTF-8)

Linked service: ls_dl_covidreportingdl

Object: SalesTransaction.csv

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity
1	581482	12/9/2019	22485	Set Of 2 Wooden Market Crates	21.47	12
2	581475	12/9/2019	22596	Christmas Star Wish List Chalkboard	10.65	36
3	581475	12/9/2019	23235	Storage Tin Vintage Leaf	11.53	12
4	581475	12/9/2019	23272	Tree T-Light Holder Willie Winkie	10.65	12

Figure 1: Ingesting Business Transaction CSV Dataset to Data Lake



api.bnm.gov.my/public/exchange-rate

Page Not Found

Sorry, the page you are looking for could not be found.

© 2018 - 2023 Bank Negara Malaysia

REST
exchange_rest

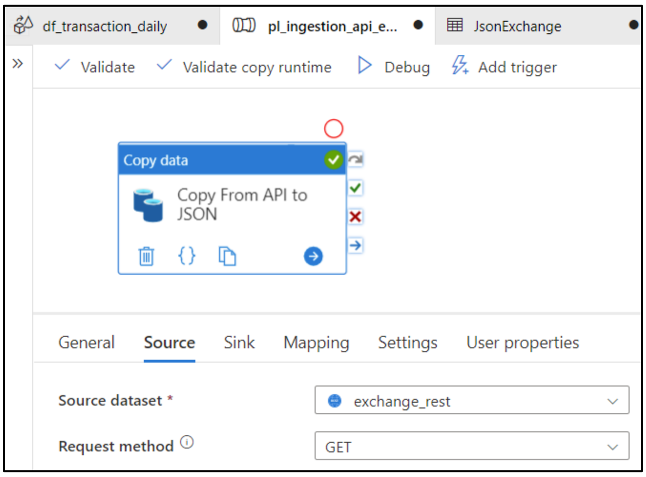
Linked service * azure_rest

Connection successful

Base URL: https://api.exchangerate.host/latest

Relative URL:

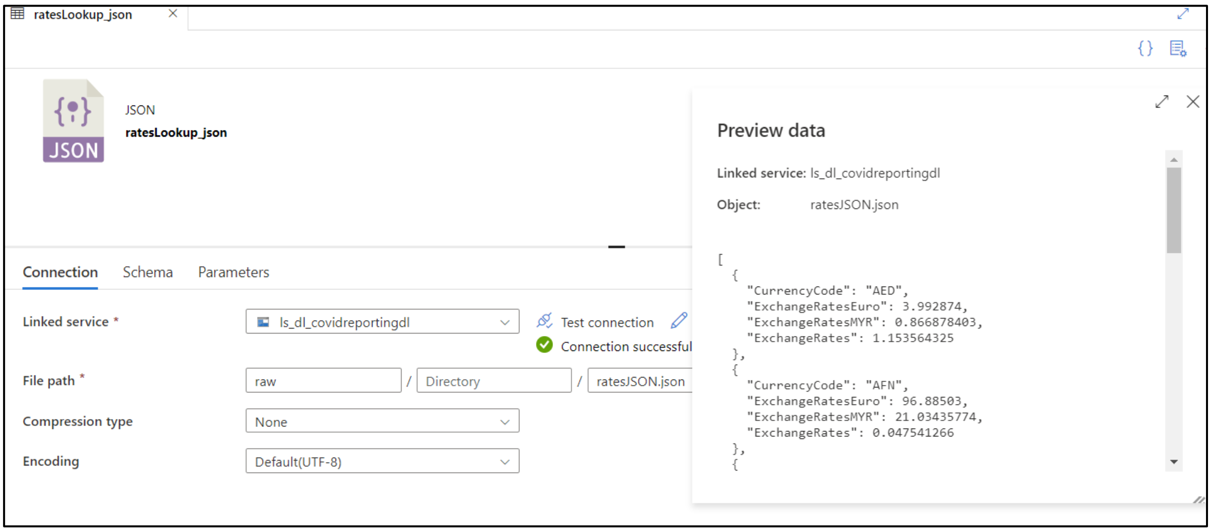
Figure 2: Setting up a REST Linked Service to the Exchange Rates Dataset API URL



Source dataset * exchange_rest

Request method GET

Figure 3: Copy data from REST Linked Service into Data Lake



Linked service * ls_dl_covidreportingdl

Connection successful

File path * raw / Directory / ratesJSON.json

Compression type None

Encoding Default(UTF-8)

Linked service: ls_dl_covidreportingdl

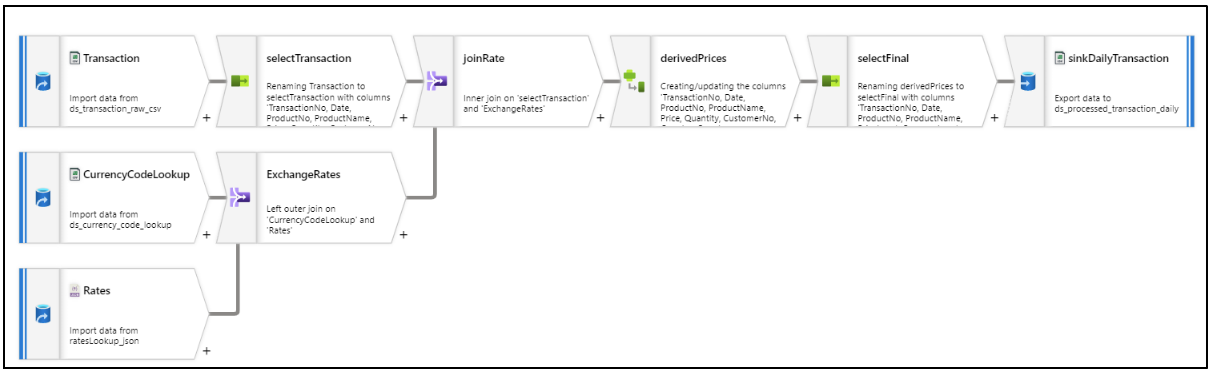
Object: ratesJSON.json

```
[
  {
    "CurrencyCode": "AED",
    "ExchangeRatesEuro": 3.992874,
    "ExchangeRatesMYR": 0.866878403,
    "ExchangeRates": 1.153564325
  },
  {
    "CurrencyCode": "AFN",
    "ExchangeRatesEuro": 96.88503,
    "ExchangeRatesMYR": 21.03435774,
    "ExchangeRates": 0.047541266
  },
  {
```

Figure 4: Successfully Ingested Exchange Rates JSON Dataset into Data Lake

Data Flow Pipeline

Figure 5 display the data flow of daily transaction. First, we add 3 dataset sources from ingested data lake into the data flow (“Transaction”, “CurrencyCodeLookup” and “Rates”). We join currency code lookup with rates (“ExchangeRates”) using left outer join to map country field to the respective currency code. Then, for transaction source, we selected only the needed columns (“selectTransaction”) and joined with the lookup that we created before (“joinRate”). From there we need to create 5 new columns; PriceLocal, PriceConverted, ExchangeRate, TotalPriceLocal, TotalPriceConverted. By using Derived Column function, we calculated for each new columns (Figure 6) to be derived from exchange rates information available using the expression builder. Last but not least, we selected only required columns, rearrange them in order as shown in Figure 7 and successfully sink them into a new data lake (Figure 8) as a processed dataset. Figure 9 shows a snippet of the processed dataset after exported into a csv file.



Transaction
Import data from ds_transaction_raw_csv

selectTransaction
Renaming Transaction to selectTransaction with columns 'TransactionNo, Date, ProductNo, ProductName,

joinRate
Inner join on 'selectTransaction' and 'ExchangeRates'

derivedPrices
Creating/updating the columns 'TransactionNo, Date, ProductNo, ProductName, Price, Quantity, CustomerNo,

selectFinal
Renaming derivedPrices to selectFinal with columns 'TransactionNo, Date, ProductNo, ProductName,

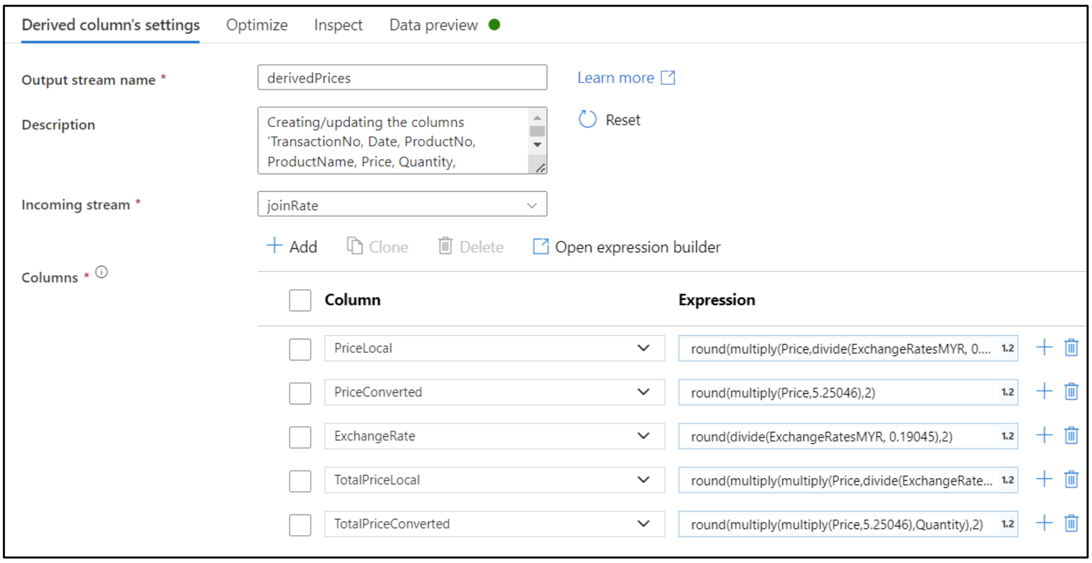
sinkDailyTransaction
Export data to ds_processed_transaction_daily

CurrencyCodeLookup
Import data from ds_currency_code_lookup

ExchangeRates
Left outer join on 'CurrencyCodeLookup' and 'Rates'

Rates
Import data from ratesLookup_json

Figure 5: Overall Data Flow of Daily Transaction



Output stream name * derivedPrices

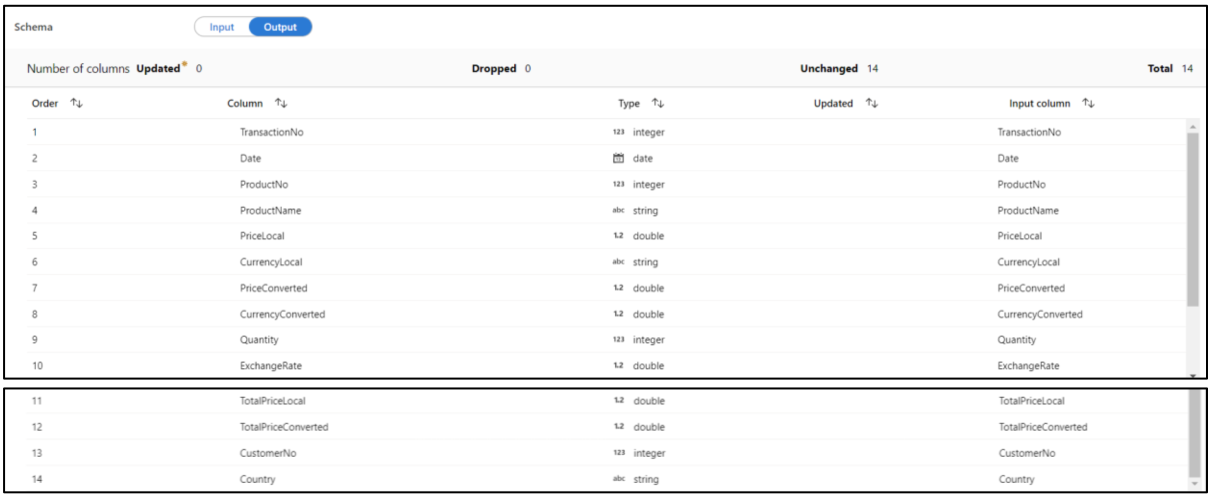
Description: Creating/updating the columns 'TransactionNo, Date, ProductNo, ProductName, Price, Quantity,

Incoming stream * joinRate

Columns *

Column	Expression
PriceLocal	round(multiply(Price,divide(ExchangeRatesMYR, 0....
PriceConverted	round(multiply(Price,5.25046),2)
ExchangeRate	round(divide(ExchangeRatesMYR, 0.19045),2)
TotalPriceLocal	round(multiply(multiply(Price,divide(ExchangeRate...
TotalPriceConverted	round(multiply(multiply(Price,5.25046),Quantity),2)

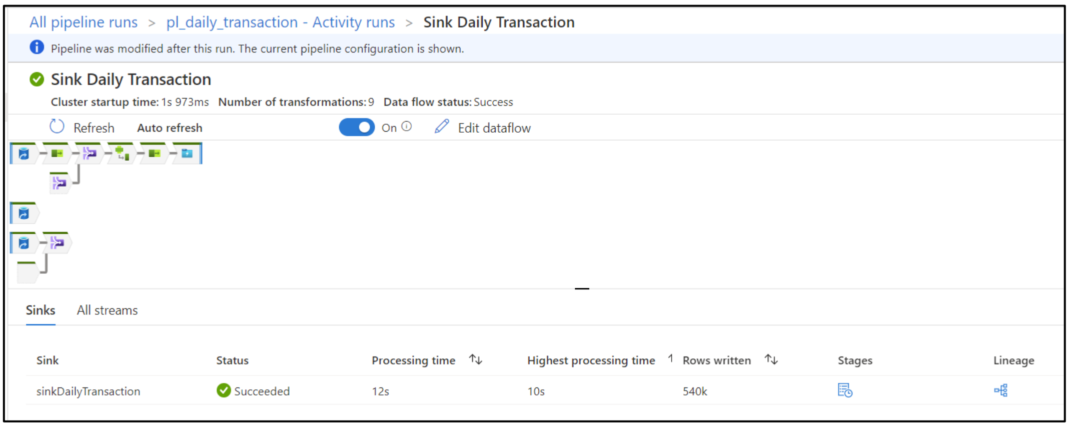
Figure 6: Creating Derived Columns According to Requirements



Number of columns Updated* 0 Dropped 0 Unchanged 14 Total 14

Order	Column	Type	Updated	Input column
1	TransactionNo	123 integer		TransactionNo
2	Date	date		Date
3	ProductNo	123 integer		ProductNo
4	ProductName	abc string		ProductName
5	PriceLocal	1.2 double		PriceLocal
6	CurrencyLocal	abc string		CurrencyLocal
7	PriceConverted	1.2 double		PriceConverted
8	CurrencyConverted	1.2 double		CurrencyConverted
9	Quantity	123 integer		Quantity
10	ExchangeRate	1.2 double		ExchangeRate
11	TotalPriceLocal	1.2 double		TotalPriceLocal
12	TotalPriceConverted	1.2 double		TotalPriceConverted
13	CustomerNo	123 integer		CustomerNo
14	Country	abc string		Country

Figure 7: Final Transformation Output for Daily Transaction



All pipeline runs > pl_daily_transaction - Activity runs > Sink Daily Transaction

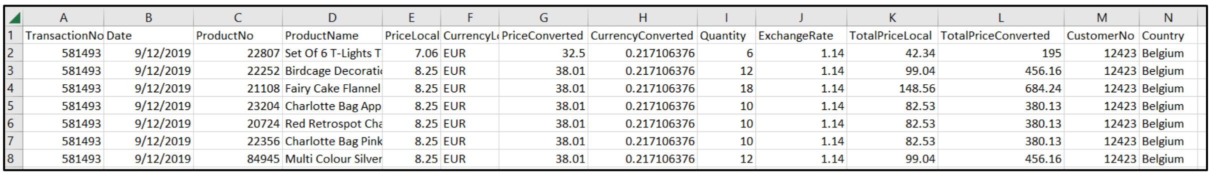
Pipeline was modified after this run. The current pipeline configuration is shown.

Sink Daily Transaction

Cluster startup time: 1s 973ms Number of transformations: 9 Data flow status: Success

Sink	Status	Processing time	Highest processing time	Rows written	Stages	Lineage
sinkDailyTransaction	✓ Succeeded	12s	10s	540k		

Figure 8: Successfully Sink the Daily Transaction Pipeline

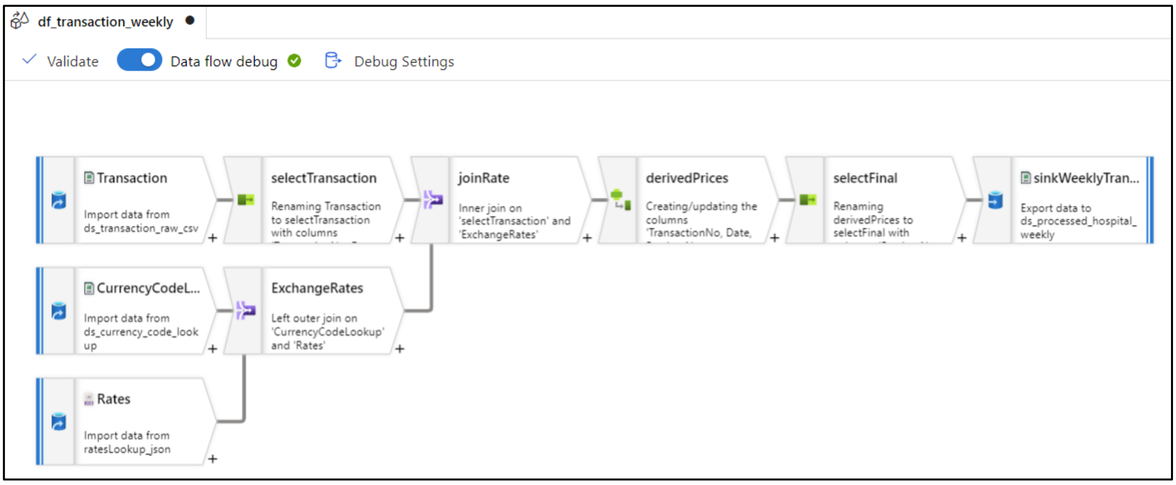


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	TransactionNo	Date	ProductNo	ProductName	PriceLocal	CurrencyL	PriceConverted	CurrencyConverted	Quantity	ExchangeRate	TotalPriceLocal	TotalPriceConverted	CustomerNo	Country
2	581493	9/12/2019	22807	Set Of 6 T-Lights T	7.06	EUR	32.5	0.217106376	6	1.14	42.34	195	12423	Belgium
3	581493	9/12/2019	22252	Birdcage Decorati	8.25	EUR	38.01	0.217106376	12	1.14	99.04	456.16	12423	Belgium
4	581493	9/12/2019	21108	Fairy Cake Flannel	8.25	EUR	38.01	0.217106376	18	1.14	148.56	684.24	12423	Belgium
5	581493	9/12/2019	23204	Charlotte Bag App	8.25	EUR	38.01	0.217106376	10	1.14	82.53	380.13	12423	Belgium
6	581493	9/12/2019	20724	Red Retrospot Cha	8.25	EUR	38.01	0.217106376	10	1.14	82.53	380.13	12423	Belgium
7	581493	9/12/2019	22356	Charlotte Bag Pink	8.25	EUR	38.01	0.217106376	10	1.14	82.53	380.13	12423	Belgium
8	581493	9/12/2019	84945	Multi Colour Silver	8.25	EUR	38.01	0.217106376	12	1.14	99.04	456.16	12423	Belgium

Figure 9: Sample of Exported Sink Daily Transaction File

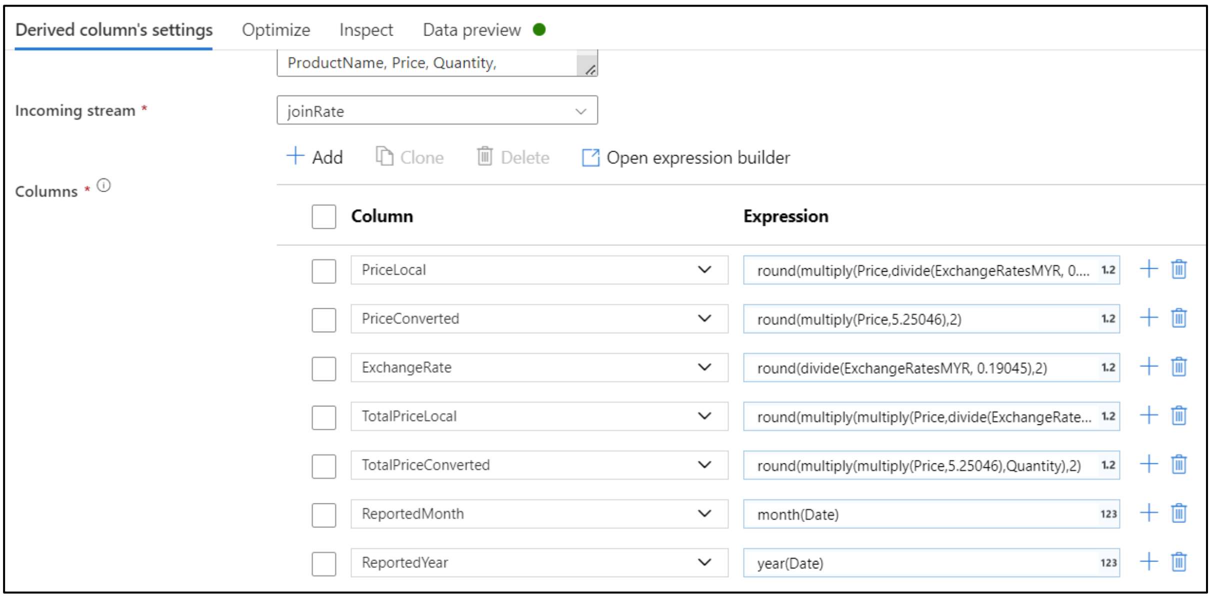
Weekly Transaction:

For weekly transaction, the process is similar with the daily transaction with added derived columns (reportedMonth and reportedYear) and removal of some columns. Figure 10 shows the overall data flow of weekly transaction. We created new calculations in the derivedPrices to derive reportedMonth and reportedYear using the expression builder to get the month and the year of transaction respectively. The rest of the process flow is similar to daily transaction where we selected only the required columns (Figure 12) and successfully sink the processed dataset into a new data lake (Figure 13). Figure 14 shows a snippet of the processed dataset for weekly transaction after exported into a csv file.



df_transaction_weekly
Validate
Data flow debug
Debug Settings
Transaction
Import data from ds_transaction_raw_csv
selectTransaction
Renaming Transaction to selectTransaction with columns
joinRate
Inner join on 'selectTransaction' and 'ExchangeRates'
derivedPrices
Creating/updating the columns 'TransactionNo, Date,
selectFinal
Renaming derivedPrices to selectFinal with
sinkWeeklyTran...
Export data to ds_processed_hospital_weekly
CurrencyCodeL...
Import data from ds_currency_code_lookup
ExchangeRates
Left outer join on 'CurrencyCodeLookup' and 'Rates'
Rates
Import data from ratesLookup_json

Figure 10: Overall Data Flow of Weekly Transaction



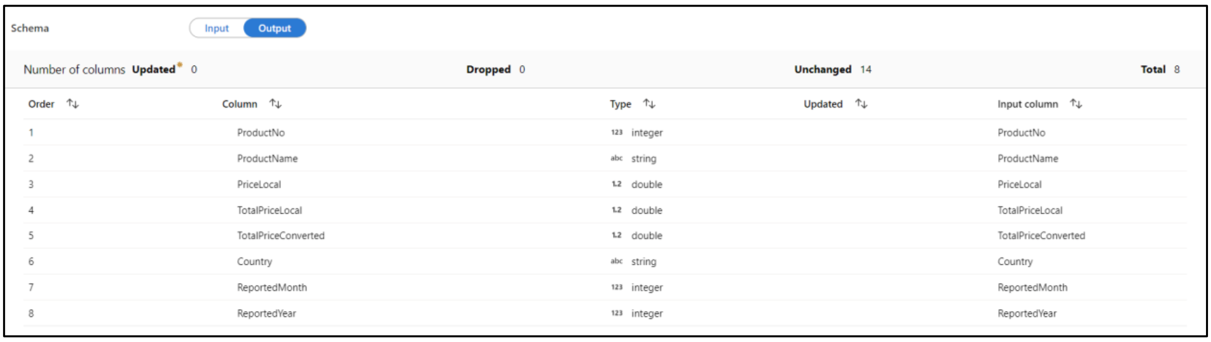
ProductName, Price, Quantity,

Incoming stream * joinRate

Columns *

□ Column	Expression
□ PriceLocal	round(multiply(Price,divide(ExchangeRatesMYR, 0.... 1.2
□ PriceConverted	round(multiply(Price,5.25046),2) 1.2
□ ExchangeRate	round(divide(ExchangeRatesMYR, 0.19045),2) 1.2
□ TotalPriceLocal	round(multiply(multiply(Price,divide(ExchangeRate... 1.2
□ TotalPriceConverted	round(multiply(multiply(Price,5.25046),Quantity),2) 1.2
□ ReportedMonth	month(Date) 123
□ ReportedYear	year(Date) 123

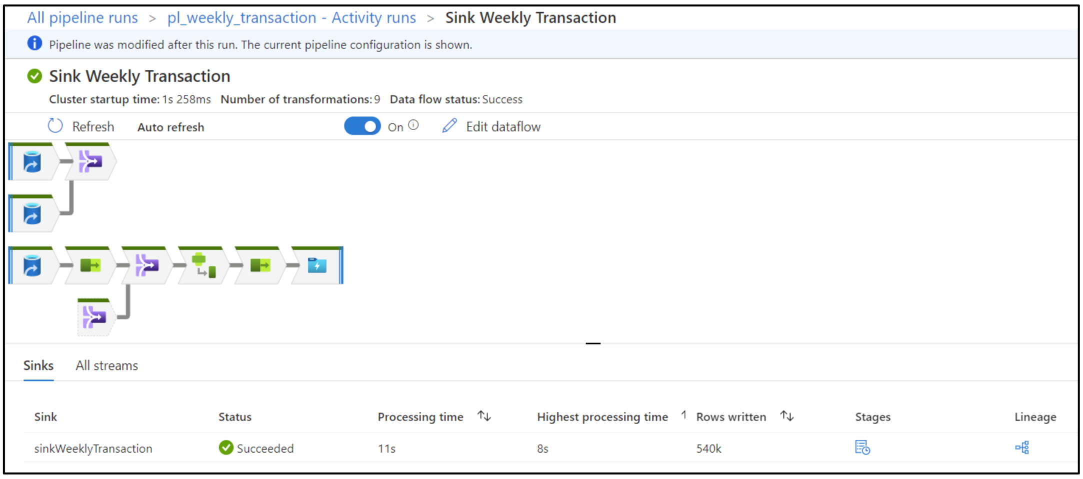
Figure 11: Creating Derived Columns According to Weekly Transaction Requirements



Number of columns **Updated*** 0 **Dropped** 0 **Unchanged** 14 **Total** 8

Order	Column	Type	Updated	Input column
1	ProductNo	123 integer		ProductNo
2	ProductName	abc string		ProductName
3	PriceLocal	1.2 double		PriceLocal
4	TotalPriceLocal	1.2 double		TotalPriceLocal
5	TotalPriceConverted	1.2 double		TotalPriceConverted
6	Country	abc string		Country
7	ReportedMonth	123 integer		ReportedMonth
8	ReportedYear	123 integer		ReportedYear

Figure 12: Final Transformation Output for Weekly Transaction



All pipeline runs > pl_weekly_transaction - Activity runs > Sink Weekly Transaction

Pipeline was modified after this run. The current pipeline configuration is shown.

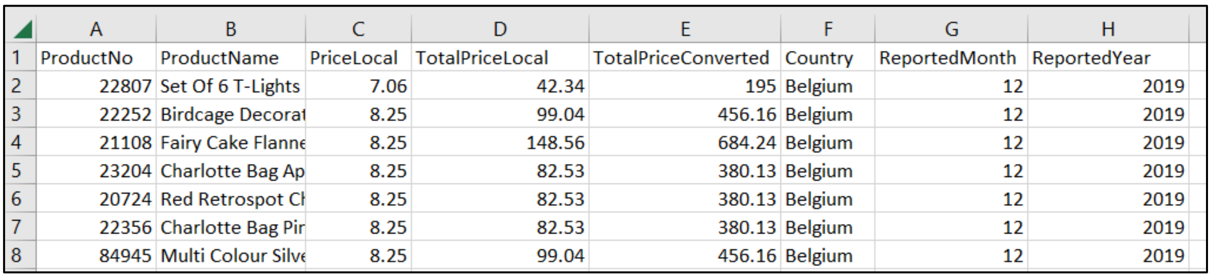
Sink Weekly Transaction

Cluster startup time: 1s 258ms **Number of transformations:** 9 **Data flow status:** Success

Refresh
Auto refresh
On
Edit dataflow
Sinks
All streams

Sink	Status	Processing time	Highest processing time	Rows written	Stages	Lineage
sinkWeeklyTransaction	✓ Succeeded	11s	8s	540k		

Figure 13: Successfully Sink the Weekly Transaction Pipeline



	A	B	C	D	E	F	G	H
1	ProductNo	ProductName	PriceLocal	TotalPriceLocal	TotalPriceConverted	Country	ReportedMonth	ReportedYear
2	22807	Set Of 6 T-Lights	7.06	42.34	195	Belgium	12	2019
3	22252	Birdcage Decorat	8.25	99.04	456.16	Belgium	12	2019
4	21108	Fairy Cake Flanne	8.25	148.56	684.24	Belgium	12	2019
5	23204	Charlotte Bag Ap	8.25	82.53	380.13	Belgium	12	2019
6	20724	Red Retrospot Cl	8.25	82.53	380.13	Belgium	12	2019
7	22356	Charlotte Bag Pir	8.25	82.53	380.13	Belgium	12	2019
8	84945	Multi Colour Silve	8.25	99.04	456.16	Belgium	12	2019

Figure 14: Sample of Exported Sink Weekly Transaction File

Summary

We are able to ingest both E-Commerce Business Transaction and Exchange Rates dataset into data lake although using different exchange rates API. Next, we are able to map Country field from transaction to respective currency code and retrieve equivalent price in MYR according to date of exchange rate and transaction. Moreover, we are able to calculate total price for each transaction in local currency and MYR and others required fields from the expected sink datasets structure for both daily and weekly transactions. Last but not least, both daily and weekly transactions successfully sink into new data lake using the explained pipelines and data flow.