Model Risk Management

Model Name

Raiffeisen Bank International

Table of Contents

Implement CDC In DLT Pipeline: Change Data Capture

Importance of Change Data Capture (CDC)

Change Data Capture (CDC) is the process that captures the changes in records made to transactional Database (Mysql, Postgre) or Data Warehouse. CDC captures operations like data deletion, append and updating, typically as a stream to re-materialize the table in external systems.

CDC enables incremental loading while eliminating the need for bulk load updating.

By capturing CDC events, we can re-materialize the source table as Delta Table in our Lakehouse and start running Analysis on top of it (Data Science, BI), merging the data with external system.

Capturing CDC

A variety of **CDC tools** are available. One of the open source leader solution is Debezium, but other implementation exists simplifying the datasource, such as Fivetran, Qlik Replicate, Streamset, Talend, Oracle GoldenGate, AWS DMS.

In this demo we are using CDC data coming from an external system like Debezium or DMS.

Debezium takes care of capturing every changed row. It typically sends the history of data changes to Kafka logs or save them as file. To simplify the demo, we'll consider that our external CDC system is up and running and saving the CDC as JSON file in our blob storage (S3, ADLS, GCS).

Our job is to CDC informations from the `customer` table (json format), making sure they're correct, and then materializing the customer table in our Lakehouse.

Materializing table from CDC events with Delta Live Table

In this example, we'll synchronize data from the Customers table in our MySQL database.

- We extract the changes from our transactional database using Debezium or any other tool and save them in a cloud object storage (S3 folder, ADLS, GCS).
- Using Autoloader we incrementally load the messages from cloud object storage, and stores the raw messages them in the `customers_cdc`. Autoloader will take care of infering the schema and handling schema evolution for us.
- Then we'll add a view `customers_cdc_clean` to check the quality of our data, using expectation, and then build dashboards to track data quality. As example the ID should never be null as we'll use it to run our upsert operations.
- Finally we perform the APPLY CHANGES INTO (doing the upserts) on the cleaned cdc data to apply the changes to the final `customers` table
- Extra: we'll also see how DLT can simply create Slowly Changing Dimention of type 2 (SCD2) to keep track of all the changes

Here is the flow we'll implement, consuming CDC data from an external database. Note that the incoming could be any format, including message queue such as Kafka.

Accessing the DLT pipeline

Your pipeline has been created! You can directly access the Delta Live Table Pipeline for CDC.

CDC input from tools like Debezium

For each change, we receive a JSON message containing all the fields of the row being updated (customer name, email, address...). In addition, we have extra metadata informations including:

- operation: an operation code, typically (DELETE, APPEND, UPDATE)
- operation_date: the date and timestamp for the record came for each operation action

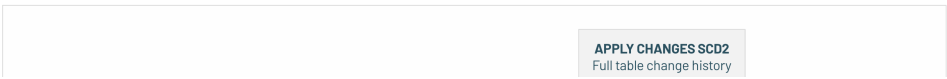
Tools like Debezium can produce more advanced output such as the row value before the change, but we'll exclude them for the clarity of the demo

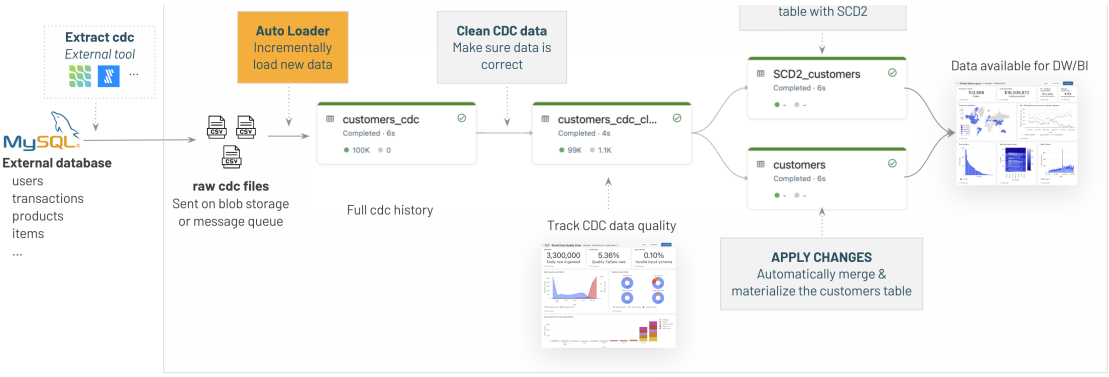
In [0]:

```
-- %python #Uncomment to explore the content
-- display(spark.read.json("/Volumes/dbdemos/dbdemos_dlt_cdc/raw_data/customers"))
```

address	email	firstname	id	lastname	operation	operation
958 Blair Drive Katherinefort, VI 31881	amandagray@spencer.com	Donald	eb174764-00af-43e7-96aa-220c15822014	Thomas	APPEND	07-04 00:
06833 Travis Village Apt. 708 New Cassandrafurt, MI 73402	juliecox@davis-cook.com	Eric	f2adb6f5-3346-4cf1-8be4-af2831b4b522	Moore	APPEND	07-04 09:
007 Steven Light Robertstown, MH 45749	pamelamccoy@thornton.com	Amy	9feffbbf-15ea-4176-b4d5-d87d7cb18e47	Payne	APPEND	07-12 16:
46383 Fisher Squares Apt. 656 Spencerborough, MS 91318	hendersonjennifer@perry.com	Sara	a64aeab4-801d-4a78-a7b5-9aeb377924db	Howe	UPDATE	07-20 19:

1/ Ingesting data with Autoloader





Our first step is to ingest the data from the cloud storage. Again, this could be from any other source like (message queue etc).

This can be challenging for multiple reason. We have to:

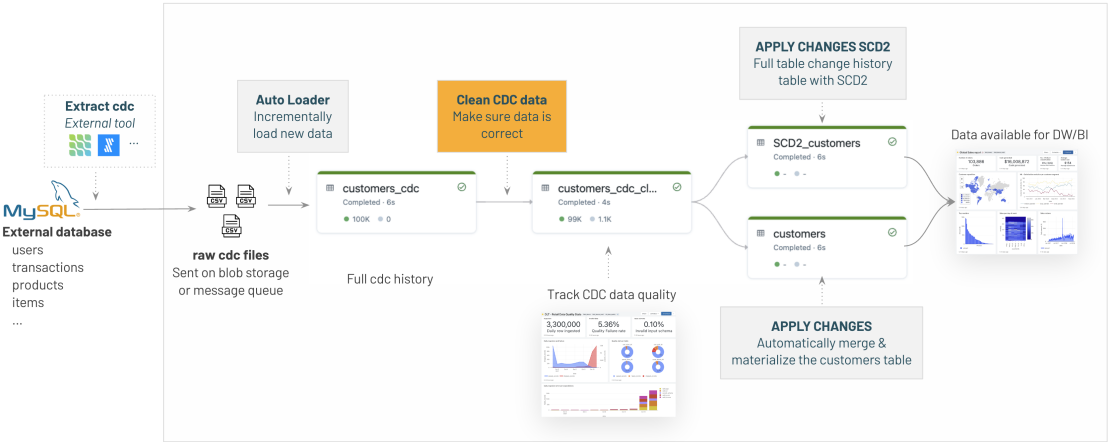
- operate at scale, potentially ingesting millions of small files
- infer schema and json type
- handle bad record with incorrect json schema
- take care of schema evolution (ex: new column in the customer table)

Databricks Autoloader solves all these challenges out of the box.

In [0]:

```
CREATE STREAMING LIVE TABLE customers_cdc
COMMENT "New customer data incrementally ingested from cloud object storage landing zone"
AS SELECT * FROM cloud_files("/Volumes/dbdemos/dbdemos_dlt_cdc/raw_data/customers", "json", map("cloud
Files.inferColumnTypes", "true"));
```

2/ Cleanup & expectations to track data quality



Next, we'll add expectations to controle data quality. To do so, we'll create a view (we don't need to duplicate the data) and check the following conditions:

- ID must never be null
- the cdc operation type must be valid
- the json must have been properly read by the autoloader

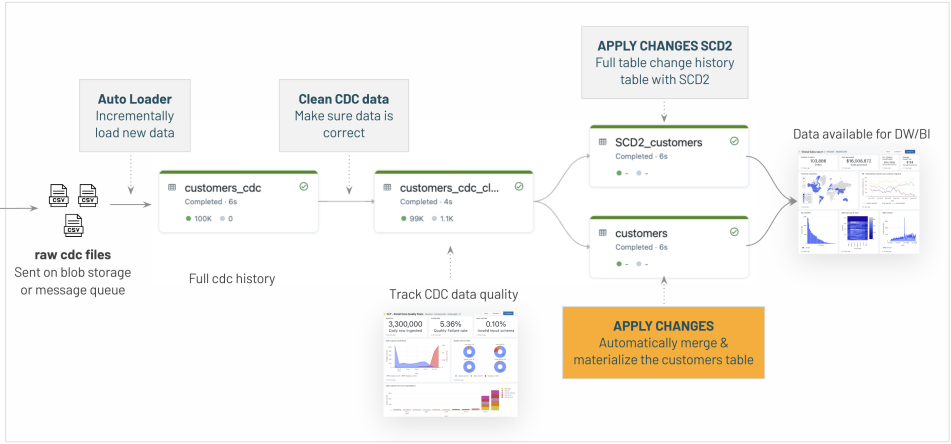
If one of these conditions isn't respected, we'll drop the row.

These expectations metrics are saved as technical tables and can then be re-used with Databricks SQL to track data quality over time.

In [0]:

```
-- this could also be a VIEW
CREATE STREAMING LIVE TABLE customers_cdc_clean(
  CONSTRAINT valid_id EXPECT (id IS NOT NULL) ON VIOLATION DROP ROW,
  CONSTRAINT valid_operation EXPECT (operation IN ('APPEND', 'DELETE', 'UPDATE')) ON VIOLATION DRO
P ROW,
  CONSTRAINT valid_json_schema EXPECT (_rescued_data IS NULL) ON VIOLATION DROP ROW
)
COMMENT "Cleansed cdc data, tracking data quality with a view. We ensude valid JSON, id and operation type"
AS SELECT *
FROM STREAM(live.customers_cdc);
```

3/ Materializing the silver table with APPLY CHANGES



The silver `customer` table will contains the most up to date view. It'll be a replicate of the original table.

This is non trivial to implement manually. You need to consider things like data deduplication to keep the most recent row.

Thanksfully Delta Live Table solve theses challenges out of the box with the `APPLY CHANGE` operation

In [0]:

```
CREATE INCREMENTAL LIVE TABLE customers
  COMMENT "Clean, materialized customers";
```

In [0]:

```
APPLY CHANGES INTO live.customers
FROM stream(live.customers_cdc_clean)
  KEYS (id)
  APPLY AS DELETE WHEN operation = "DELETE"
  SEQUENCE BY operation_date --primary key, auto-incrementing ID of any kind that can be used to identity orde
r of events, or timestamp
  COLUMNS * EXCEPT (operation, operation_date, _rescued_data);
```

4/ Slowly Changing Dimention of type 2 (SCD2)

Why SCD2

It's often required to create a table tracking all the changes resulting from APPEND, UPDATE and DELETE:

- History: you want to keep an history of all the changes from your table
- Traceability: you want to see which operation

SCD2 with DLT

Delta support CDF (Change Data Flow) and `table_change` can be used to query the table modification in a SQL/python. However, CDF main use-case is to capture changes in a pipeline and not create a full view of the table changes from the begining.

Things get especially complex to implement if you have out of order events. If you need to sequence your changes by a timestamp and receive a modification which happened in the past, then you not only need to append a new entry in your SCD table, but also update the previous entries.

Delta Live Table makes all this logic super simple and let you create a separate table containing all the modifications, from the begining of the time. This table can then be used at scale, with specific partitions / zorder columns if required. Out of order fields will be handled out of the box based on the _sequence_by

To create a SCD2 table, all we have to do is leverage the `APPLY CHANGES` with the extra option: `STORED AS {SCD TYPE 1 | SCD TYPE 2 [WITH {TIMESTAMP|VERSION}}]`

*Note: you can also limit the columns being tracked with the option: `TRACK HISTORY ON {columnList | EXCEPT(exceptColumnList)}**

In [0]:

```
-- create the table
CREATE INCREMENTAL LIVE TABLE SCD2_customers
  COMMENT "Slowly Changing Dimension Type 2 for customers";

-- store all changes as SCD2
APPLY CHANGES INTO live.SCD2_customers
FROM stream(live.customers_cdc_clean)
  KEYS (id)
  APPLY AS DELETE WHEN operation = "DELETE"
  SEQUENCE BY operation_date
  COLUMNS * EXCEPT (operation, operation_date, _rescued_data)
  STORED AS SCD TYPE 2 ;
```

Conclusion

We now have our DLT pipeline up & ready! Our `customers` table is materialize and we can start building BI report

to analyze and improve our business. It also open the door to Data Science and ML use-cases such as customer churn, segmentation etc.

Monitoring your data quality metrics with Delta Live Table

Delta Live Tables tracks all your data quality metrics. You can leverage the expecations directly as SQL table with Databricks SQL to track your expectation metrics and send alerts as required.

This let you build custom dashboards to track those metrics.

Data Quality Dashboard

For more detail on how to analyse Expectation metrics, open the 03-Retail_DLT_CDC_Monitoring notebook.

Implement CDC In DLT Pipeline: Change Data Capture

Importance of Change Data Capture (CDC)

Change Data Capture (CDC) is the process that captures the changes in records made to transactional Database (Mysql, Postgre) or Data Warehouse. CDC captures operations like data deletion, append and updating, typically as a stream to re-materialize the table in external systems.

CDC enables incremental loading while eliminating the need for bulk load updating.

By capturing CDC events, we can re-materialize the source table as Delta Table in our Lakehouse and start running Analysis on top of it (Data Science, BI), merging the data with external system.

Capturing CDC

A variety of **CDC tools** are available. One of the open source leader solution is Debezium, but other implementation exists simplifying the datasource, such as Fivetran, Qlik Replicate, Streamset, Talend, Oracle GoldenGate, AWS DMS.

In this demo we are using CDC data coming from an external system like Debezium or DMS.

Debezium takes care of capturing every changed row. It typically sends the history of data changes to Kafka logs or save them as file. To simplify the demo, we'll consider that our external CDC system is up and running and saving the CDC as JSON file in our blob storage (S3, ADLS, GCS).

Our job is to CDC informations from the `customer` table (json format), making sure they're correct, and then materializing the customer table in our Lakehouse.

Materializing table from CDC events with Delta Live Table

In this example, we'll synchronize data from the Customers table in our MySQL database.

- We extract the changes from our transactional database using Debezium or any other tool and save them in a cloud object storage (S3 folder, ADLS, GCS).
- Using Autoloader we incrementally load the messages from cloud object storage, and stores the raw messages them in the `customers_cdc`. Autoloader will take care of infering the schema and handling schema evolution for us.
- Then we'll add a view `customers_cdc_clean` to check the quality of our data, using expectation, and then build dashboards to track data quality. As example the ID should never be null as we'll use it to run our upsert operations.
- Finally we perform the APPLY CHANGES INTO (doing the upserts) on the cleaned cdc data to apply the changes to the final `customers` table
- Extra: we'll also see how DLT can simply create Slowly Changing Dimention of type 2 (SCD2) to keep track of all the changes

Here is the flow we'll implement, consuming CDC data from an external database. Note that the incoming could be any format, including message queue such as Kafka.

Accessing the DLT pipeline

Your pipeline has been created! You can directly access the Delta Live Table Pipeline for CDC.

CDC input from tools like Debezium

For each change, we receive a JSON message containing all the fields of the row being updated (customer name, email, address...). In addition, we have extra metadata informations including:

- operation: an operation code, typically (DELETE, APPEND, UPDATE)
- operation_date: the date and timestamp for the record came for each operation action

Tools like Debezium can produce more advanced output such as the row value before the change, but we'll exclude them for the clarity of the demo

In [0]:

```
%sql
-- %python #Uncomment to explore the content
-- display(spark.read.json("/Volumes/dbdemos/dbdemos_dlt_cdc/raw_data/customers"))
```

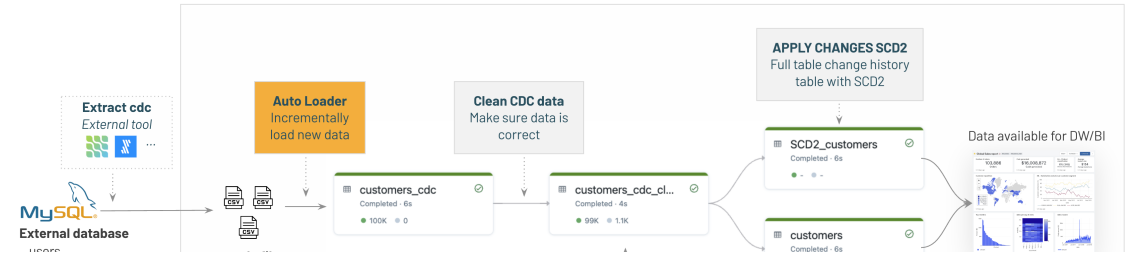
CDC with DLT & Python Syntax

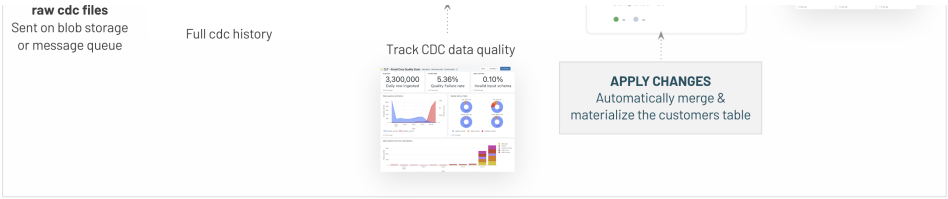
DLT tables, views, and their associated settings are configured using decorators.

If you're unfamiliar with Python decorators, just note that they are functions or classes preceded with the `@` sign that interact with the next function present in a Python script.

The `@dlt.table` decorator is the basic method for turning a Python function into a Delta Live Table.

1/ Ingesting data with Autoloader





Our first step is to ingest the data from the cloud storage. Again, this could be from any other source like (message queue etc).

This can be challenging for multiple reason. We have to:

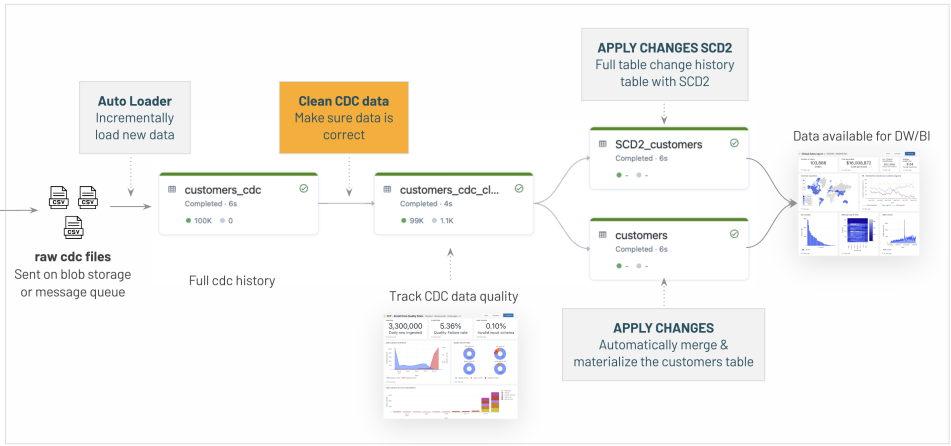
- operate at scale, potentially ingesting millions of small files
- infer schema and json type
- handle bad record with incorrect json schema
- take care of schema evolution (ex: new column in the customer table)

Databricks Autoloader solves all these challenges out of the box.

In [0]:

```
import dlt
from pyspark.sql.functions import *
##Create the bronze information table containing the raw JSON data taken from the storage path printed in Cmd5 i
n 00_Retail_Data_CDC_Generator notebook
@dlt.create_table(name="customers_cdc",
                  comment = "New customer data incrementally ingested from cloud object storage landing zone")
def customers_cdc():
  return (
    spark.readStream.format("cloudFiles")
      .option("cloudFiles.format", "json")
      .option("cloudFiles.inferColumnTypes", "true")
      .load("/Volumes/dbdemos/dbdemos_dlt_cdc/raw_data/customers"))
```

2/ Cleanup & expectations to track data quality



Next, we'll add expectations to controle data quality. To do so, we'll create a view (we don't need to duplicate the data) and check the following conditions:

- ID must never be null
- the cdc operation type must be valid
- the json must have been properly read by the autoloader

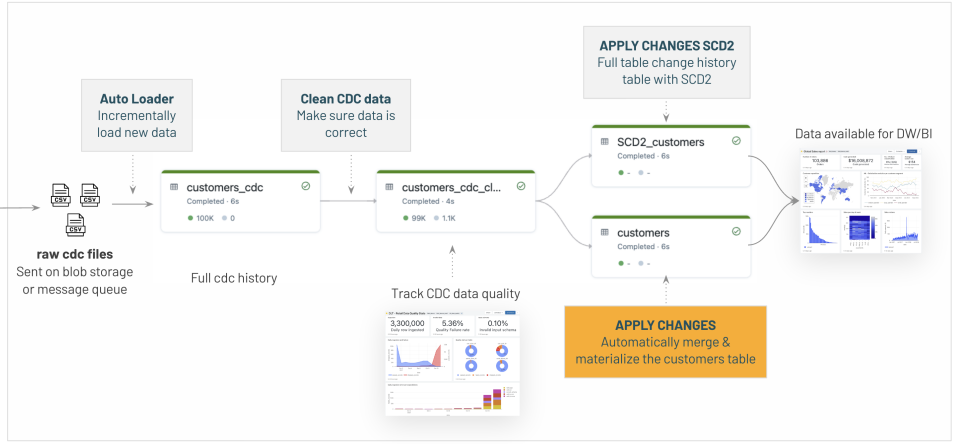
If one of these conditions isn't respected, we'll drop the row.

These expectations metrics are saved as technical tables and can then be re-used with Databricks SQL to track data quality over time.

In [0]:

```
#This could also be a view: create_view
@dlt.create_table(name="customers_cdc_clean",
                  comment="Cleansed cdc data, tracking data quality with a view. We ensude valid JSON, id and operati
on type")
@dlt.expect_or_drop("no_rescued_data", "_rescued_data IS NULL")
@dlt.expect_or_drop("valid_id", "id IS NOT NULL")
@dlt.expect_or_drop("valid_operation", "operation IN ('APPEND', 'DELETE', 'UPDATE')")
def customers_cdc_clean():
  return dlt.read_stream("customers_cdc") \
          .select("address", "email", "id", "firstname", "lastname", "operation", "operation_date", "_rescued_data")
```

3/ Materializing the silver table with APPLY CHANGES



The silver `customer` table will contains the most up to date view. It'll be a replicate of the original table.

This is non trivial to implement manually. You need to consider things like data deduplication to keep the most recent row.

Thanksfully Delta Live Table solve theses challenges out of the box with the `APPLY CHANGE` operation

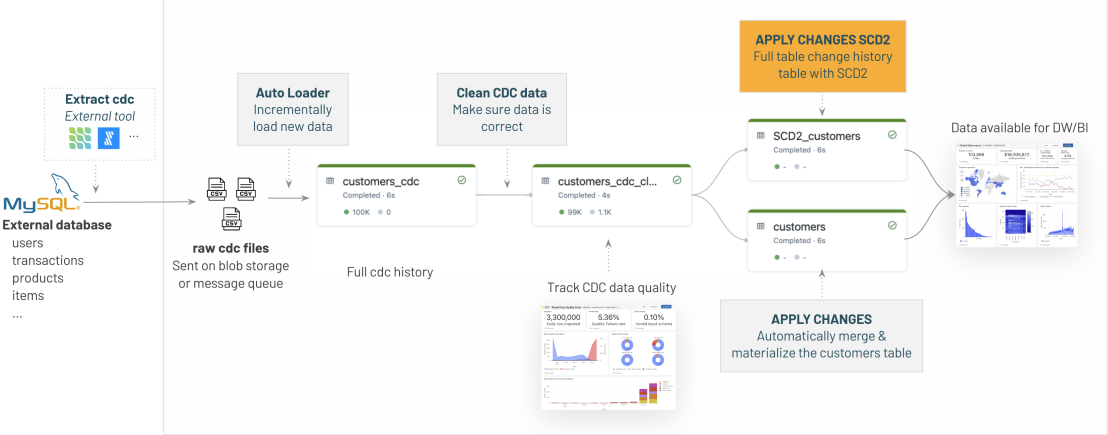
In [0]:

```
dlt.create_target_table(name="customers", comment="Clean, materialized customers")
```

In [0]:

```
dlt.apply_changes(
  target = "customers", #The customer table being materilized
  source = "customers_cdc_clean", #the incoming CDC
  keys = ["id"], #what we'll be using to match the rows to upsert
  sequence_by = col("operation_date"), #we deduplicate by operation date getting the most recent value
  ignore_null_updates = False,
  apply_as_deletes = expr("operation = 'DELETE'"), #DELETE condition
  except_column_list = ["operation", "operation_date", "_rescued_data"]) #in addition we drop metadata columns
```

4/ Slowly Changing Dimention of type 2 (SCD2)



Why SCD2

It's often required to create a table tracking all the changes resulting from APPEND, UPDATE and DELETE:

- History: you want to keep an history of all the changes from your table
- Traceability: you want to see which operation

SCD2 with DLT

Delta support CDF (Change Data Flow) and `table_change` can be used to query the table modification in a SQL/python. However, CDF main use-case is to capture changes in a pipeline and not create a full view of the table changes from the begining.

Things get especially complex to implement if you have out of order events. If you need to sequence your changes by a timestamp and receive a modification which happened in the past, then you not only need to append a new entry in your SCD table, but also update the previous entries.

Delta Live Table makes all this logic super simple and let you create a separate table containing all the modifications, from the begining of the time. This table can then be used at scale, with specific partitions / zorder columns if required. Out of order fields will be handled out of the box based on the _sequence_by

To create a SCD2 table, all we have to do is leverage the `APPLY CHANGES` with the extra option: `STORED AS {SCD TYPE 1 | SCD TYPE 2 [WITH {TIMESTAMP|VERSION}}]`

*Note: you can also limit the columns being tracked with the option: `TRACK HISTORY ON {columnList | EXCEPT(exceptColumnList)}**

In [0]:

```
#create the table
dlt.create_target_table(name="SCD2_customers", comment="Slowly Changing Dimension Type 2 for customers")

#store all changes as SCD2
dlt.apply_changes(
  target = "SCD2_customers",
  source = "customers_cdc_clean",
  keys = ["id"],
  sequence_by = col("operation_date"),
  ignore_null_updates = False,
  apply_as_deletes = expr("operation = 'DELETE'"),
  except_column_list = ["operation", "operation_date", "_rescued_data"],
  stored_as_scd_type = "2") #Enable SCD2 and store individual updates
```

Conclusion

We now have our DLT pipeline up & ready! Our `customers` table is materialize and we can start building BI report to analyze and improve our business. It also open the door to Data Science and ML use-cases such as customer churn, segmentation etc.

Monitoring your data quality metrics with Delta Live Table

Delta Live Tables tracks all your data quality metrics. You can leverage the expecations directly as SQL table with Databricks SQL to track your expectation metrics and send alerts as required.

This let you build custom dashboards to track those metrics.

Data Quality Dashboard

For more detail on how to analyse Expectation metrics, open the 03-Retail_DLT_CDC_Monitoring notebook.

Delta Live Tables - Monitoring

Each DLT Pipeline saves events and expectations metrics in the Storage Location defined on the pipeline. From this table we can see what is happening and the quality of the data passing through it.

You can leverage the expecations directly as a SQL table with Databricks SQL to track your expectation metrics and send alerts as required.

This notebook extracts and analyses expectation metrics to build such KPIS.

Accessing the Delta Live Table pipeline events with Unity Catalog

Databricks provides an `event_log` function which is automatically going to lookup the event log table. You can specify any table to get access to the logs:

`SELECT * FROM event_log(TABLE(catalog.schema.my_table))`

Using Legacy hive_metastore

Note: If you are not using Unity Catalog (legacy hive_metastore), you can find your event log location opening the Settings of your DLT pipeline, under `storage` :

```
{
    ...
    "name": "lakehouse_churn_dlt",
    "storage": "/demos/dlt/loans",
    "target": "your schema"
}
```

A cluster has been created for this demo

To run this demo, just select the cluster `dbdemos-dlt-cdc-erika_fonseca` from the dropdown menu (open cluster configuration).
Note: If the cluster was deleted after 30 days, you can re-create it with `dbdemos.create_cluster('dlt-cdc')` or re-install the demo: `dbdemos.install('dlt-cdc')`

In [0]:

```
%sql
SELECT * FROM event_log(TABLE(dbdemos.dbdemos_dlt_cdc.customers))
```

id id	sequence sequence	origin origin	timestamp timestamp
6f05d5c0-4523-11ef-807d-00163e2d74b9	List(List(execution, 1721320390162001), null)	List(AWS, us-west-2, 1660015457675682, null, fe1fdf40-ed56-48f8-bfb4-29f20e2e7336, WORKSPACE, dbdemos_dlt_cdc_build, 0718-162907-z5wx0ea3, 37e71105-8a45-42ec-9cb0-496349b0ffbb, null, null, null, null, null, 37e71105-8a45-42ec-9cb0-496349b0ffbb, fe1fdf40-ed56-48f8-bfb4-29f20e2e7336)	2024-07-18T16:33:16.06Z

System table setup

We'll create a table based on the events log being saved by DLT. The system tables are stored under the storage path defined in your DLT settings (the one defined in the widget):

In [0]:

```
%sql
CREATE OR REPLACE TEMPORARY VIEW demo_cdc_dlt_system_event_log_raw
  as SELECT * FROM event_log(TABLE(dbdemos.dbdemos_dlt_cdc.customers));
SELECT * FROM demo_cdc_dlt_system_event_log_raw order by timestamp desc;
```

id	sequence	origin	timestamp
ea3b2a30-480f-11ef-bc91-00163ee03692	List(List(execution, 1721640848138067), null)	List(AWS, us-west-2, 1660015457675682, null, fe1fdf40-ed56-48f8-bfb4-29f20e2e7336, WORKSPACE, dbdemos_dlt_cdc_build, 0722-093018-4g3pv730, 690dc21b-df39-47f2-86d6-c487b6a5b008, null, null, null, null, null, 690dc21b-df39-47f2-86d6-c487b6a5b008, fe1fdf40-ed56-48f8-bfb4-29f20e2e7336)	2024-07-22T09:51:06.323Z
c677e430-480f-11ef-bc91-00163ee03692	List(List(execution, 1721640848138066), null)	List(AWS, us-west-2, 1660015457675682, null, fe1fdf40-ed56-48f8-bfb4-29f20e2e7336, WORKSPACE, dbdemos_dlt_cdc_build, 0722-093018-4g3pv730, 690dc21b-df39-47f2-86d6-c487b6a5b008, null, null, null, null, null, 690dc21b-df39-47f2-86d6-c487b6a5b008, fe1fdf40-ed56-48f8-bfb4-29f20e2e7336)	2024-07-22T09:50:06.323Z

Delta Live Table expectation analysis

Delta live table tracks our data quality through expectations. These expectations are stored as technical tables without the DLT log events. We can create a view to simply analyze this information

Make sure you set your DLT storage path in the widget!

Analyzing dlt_system_event_log_raw table structure

The `details` column contains metadata about each Event sent to the Event Log. There are different fields depending on what type of Event it is. Some examples include:

- `user_action` Events occur when taking actions like creating the pipeline
- `flow_definition` Events occur when a pipeline is deployed or updated and have lineage, schema, and execution plan information
  - `output_dataset` and `input_datasets` - output table/view and its upstream table(s)/view(s)
  - `flow_type` - whether this is a complete or append flow
  - `explain_text` - the Spark explain plan
- `flow_progress` Events occur when a data flow starts running or finishes processing a batch of data
  - `metrics` - currently contains `num_output_rows`
  - `data_quality` - contains an array of the results of the data quality rules for this particular dataset
    - `dropped_records`
    - `expectations`
      - `name`, `dataset`, `passed_records`, `failed_records`

We can leverage this information to track our table quality using SQL

In [0]:

```
%sql
SELECT
    id,
    timestamp,
    sequence,
    event_type,
    message,
    level,
    details
 FROM demo_cdc_dlt_system_event_log_raw
ORDER BY timestamp ASC;
```

id	timestamp	sequence	event_type	
d89eb8e0-4522-11ef-85ae-e272fbc424e1	2024-07-18T16:29:03.726Z	List(List(execution, 1721320390162006), 1721320143728001)	user_action	User quentin.ambard@databricks.com (
dad68070-4522-11ef-85ae-e272fbc424e1	2024-07-18T16:29:07.447Z	List(List(execution, 1721320390162007), 1721320147448001)	user_action	User quentin.ambard@databricks.com st
dadb1450-4522-11ef-85ae-e272fbc424e1	2024-07-18T16:29:07.477Z	List(List(execution, 1721320390162008), 1721320147478001)	create_update	Update 37e711 starte
db17a820-4522-11ef-85ae-	2024-07-18T16:29:07.874Z	List(List(execution, 1721320390162009), 1721320147875001)	update_progress	Update 37e711 is WAITING_FOR

In [0]:

```
%sql
create or replace temp view cdc_dlt_expectations as (
  SELECT
    id,
    timestamp,
    details:flow_progress.metrics.num_output_rows as output_records,
    details:flow_progress.data_quality.dropped_records,
    details:flow_progress.status as status_update,
    explode(from_json(details:flow_progress.data_quality.expectations
```

```
        ,'array<struct<dataset: string, failed_records: bigint, name: string, passed_records: bigint>>')) expectations
  FROM demo_cdc_dlt_system_event_log_raw
  where details:flow_progress.data_quality.expectations is not null
  ORDER BY timestamp);
select * from cdc_dlt_expectations
```

id	timestamp	output_records	dropped_records	status_update	expectations
b2dae6c0-4779-11ef-b958-00163e86270d	2024-07-21T15:55:48.908Z	96702	3298	RUNNING	List(customers_cdc_clean, 1300, valid_operation, 98700)
b2dae6c0-4779-11ef-b958-00163e86270d	2024-07-21T15:55:48.908Z	96702	3298	RUNNING	List(customers_cdc_clean, 0, valid_json_schema, 100000)
b2dae6c0-4779-11ef-b958-00163e86270d	2024-07-21T15:55:48.908Z	96702	3298	RUNNING	List(customers_cdc_clean, 2017, valid_id, 97983)
b8828170-480d-11ef-bc91-00163ee03692	2024-07-22T09:35:23.911Z	97285	2715	RUNNING	List(customers_cdc_clean, 700, valid_operation, 99300)
b8828170-480d-11ef-bc91-00163ee03692	2024-07-22T09:35:23.911Z	97285	2715	RUNNING	List(customers_cdc_clean, 0, valid_json_schema, 100000)
b8828170-480d-11ef-	2024-07-				List(customers_cdc_clean,

3 - Visualizing the Quality Metrics

Let's run a few queries to show the metrics we can display. Ideally, we should be using Databricks SQL to create SQL Dashboard and track all the data, but for this example we'll run a quick query in the dashboard directly:

In [0]:

```
%sql
select sum(expectations.failed_records) as failed_records, sum(expectations.passed_records) as passed_record
s, expectations.name from cdc_dlt_expectations group by expectations.name
```

failed_records	passed_records	name
2000	198000	valid_operation
0	200000	valid_json_schema
4046	195954	valid_id

Plotting failed record per expectations

In [0]:

```
import plotly.express as px
expectations_metrics = spark.sql("select sum(expectations.failed_records) as failed_records, sum(expectations.pa
ssed_records) as passed_records, expectations.name from cdc_dlt_expectations group by expectations.name").to
Pandas()
px.bar(expectations_metrics, x="name", y=["passed_records", "failed_records"], title="DLT expectations metrics")
```

What's next?

We now have our data ready to be used for more advanced.

We can start creating our first DBSQL Dashboard monitoring our data quality & DLT pipeline health.

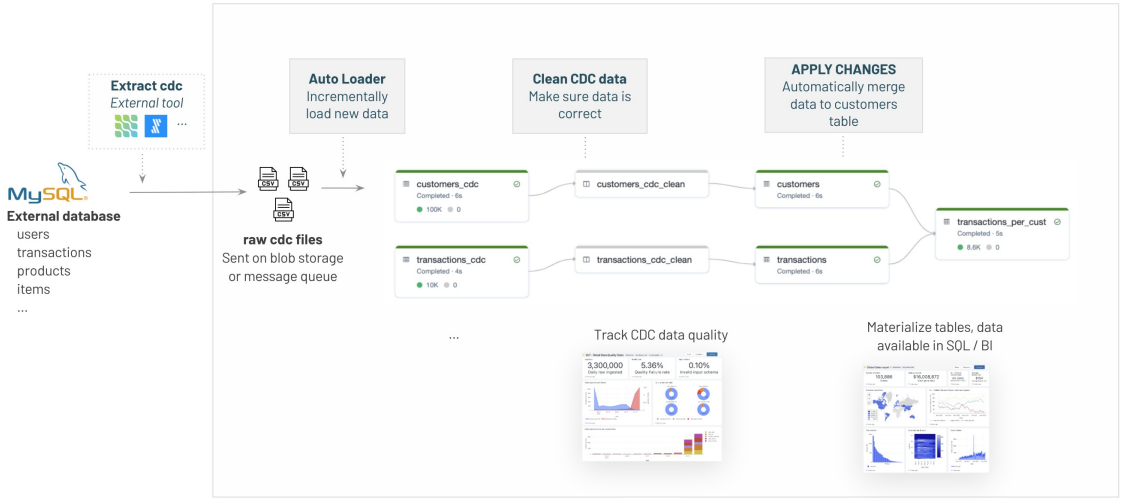
Implementing a CDC pipeline using DLT for N tables

We saw previously how to setup a CDC pipeline for a single table. However, real-life database typically involve multiple tables, with 1 CDC folder per table.

Operating and ingesting all these tables at scale is quite challenging. You need to start multiple table ingestion at the same time, working with threads, handling errors, restart where you stopped, deal with merge manually.

Thankfully, DLT takes care of that for you. We can leverage python loops to naturally iterate over the folders (see the documentation for more details)

DLT engine will handle the parallelization whenever possible, and autoscale based on your data volume.



In [0]:

```
# uncomment to see the raw files
# %fs ls /Volumes/dbdemos/dbdemos_dlt_cdc/raw_data
```

In [0]:

```
#Let's loop over all the folders and dynamically generate our DLT pipeline.
import dlt
from pyspark.sql.functions import *


def create_pipeline(table_name):
  print(f"Building DLT CDC pipeline for {table_name}")

  ##Raw CDC Table
  #       .option("cloudFiles.maxFilesPerTrigger", "1")
  @dlt.create_table(name=table_name+"_cdc",
                    comment = "New "+table_name+" data incrementally ingested from cloud object storage landing zone
")
  def raw_cdc():
    return (
      spark.readStream.format("cloudFiles")
        .option("cloudFiles.format", "json")
        .option("cloudFiles.inferColumnTypes", "true")
```

```
      .load("/Volumes/dbdemos/dbdemos_dlt_cdc/raw_data/"+table_name))

  ##Clean CDC input and track quality with expectations
  @dlt.create_view(name=table_name+"_cdc_clean",
                   comment="Cleansed cdc data, tracking data quality with a view. We ensude valid JSON, id and operation type")
  @dlt.expect_or_drop("no_rescued_data", "_rescued_data IS NULL")
  @dlt.expect_or_drop("valid_id", "id IS NOT NULL")
  @dlt.expect_or_drop("valid_operation", "operation IN ('APPEND', 'DELETE', 'UPDATE')")
  def raw_cdc_clean():
    return dlt.read_stream(table_name+"_cdc")


  ##Materialize the final table
  dlt.create_target_table(name=table_name, comment="Clean, materialized "+table_name)
  dlt.apply_changes(target = table_name, #The customer table being materilized
                    source = table_name+"_cdc_clean", #the incoming CDC
                    keys = ["id"], #what we'll be using to match the rows to upsert
                    sequence_by = col("operation_date"), #we deduplicate by operation date getting the most recent value
                    ignore_null_updates = False,
                    apply_as_deletes = expr("operation = 'DELETE'"), #DELETE condition
                    except_column_list = ["operation", "operation_date", "_rescued_data"]) #in addition we drop metadata columns


for folder in dbutils.fs.ls("/Volumes/dbdemos/dbdemos_dlt_cdc/raw_data"):
  table_name = folder.name[:-1]
  create_pipeline(table_name)
```

In [0]:

```
@dlt.create_table(name="transactions_per_customers",
                  comment = "table join between users and transactions for further analysis")
def raw_cdc():
  return dlt.read("transactions").join(dlt.read("customers"), ["id"], "left")
```

Conclusion

We can now scale our CDC pipeline to N tables using python factorization. This gives us infinite possibilities and abstraction level in our DLT pipelines.

DLT handles all the hard work for us so that we can focus on business transformation and drastically accelerate DE team:

- simplify file ingestion with the autoloader
- track data quality using exception
- simplify all operations including upsert with APPLY CHANGES
- process all our tables in parallel
- autoscale based on the amount of data

DLT gives more power to SQL-only users, letting them build advanced data pipeline without requiering strong Data Engineers skills.

Display mathematical equations

Notebooks support KaTeX for displaying mathematical formulas and equations. For example,

[Math Processing Error]

where *[Math Processing Error]*

Documentation: https://docs.databricks.com/en/notebooks/notebooks-code.html#display-mathematical-equations

LaTex

7.3 Calculation of long-run average LGD

- probability of cure (*Probability of Cure* p) is being calculated as (including open defaults based on classification described above):

Probability of Cure p = *[Math Processing Error]*

where *[Math Processing Error]* denotes indicator (characteristic) function and *[Math Processing Error]* for $\forall$ k $\in$ *calib. sample* (open and closed).

KaTeX

7.3 Calculation of long-run average LGD

- probability of cure *[Math Processing Error]* is being calculated as (including open defaults based on classification described above):

[Math Processing Error]

where *[Math Processing Error]* denotes indicator (characteristic) function and *[Math Processing Error]* for *[Math Processing Error]* (open and closed).

Table: Demo Dataset Exported as Table

Rows 0 to 29, Columns 0 to 5

customerID	first_name	last_name	email_address	phone_number	address
2000259	Kayla	Barrett	brittanyramos@example.org	349-683-9514x73065	717 Whitney Roads
2000260	Amanda	Reed	scollier@example.org	+1-999-308-9110	69075 Logan Circles Apt. 540
2000261	Steven	Tanner	haileysanchez@example.net	859-946-4140x24086	08560 Thomas Land
2000262	Jennifer	Forbes	belldonna@example.com	633-427-4977	5840 Warren Garden Suite 901
2000263	Kenneth	Berger	bdalton@example.net	(831)220-1833x906	693 Baker Dale
2000264	James	Edwards	mary39@example.org	271-321-1561x9697	665 Campbell Streets Suite 966
2000265	Ann	Montgomery	michaelreese@example.org	7673757334	220 Barber Islands Apt. 664
2000266	Heather	Moore	mannmartin@example.com	278-606-3676x938	896 Greene Hill Suite 575
2000267	Joseph	Graham	suzannereeves@example.org	538.787.9102x4963	2563 Jessica Mountains Apt. 192
2000268	Lisa	Ramirez	kristi04@example.com	(643)361-8721x12062	573 Bailey Lights Suite 941
2000269	Sierra	Jones	zgomez@example.net	(645)797-8952x96710	579 Butler Avenue Apt. 753
2000270	Tyler	Jackson	stephanieshaw@example.net	001-517-875-1644	92604 Michelle Bridge Apt. 881
2000271	Rachael	Lowe	jonathan06@example.net	+1-414-570-3089x7009	941 Carolyn Station Apt. 521
2000272	Shelly	Rogers	lnorris@example.com	584.397.7876	48472 Oconnell Shore
2000273	Heather	Greene	emilymartin@example.com	(835)366-8761	917 Tammie Fords Apt. 328
2000274	Paul	Hartman	chavezronald@example.net	(361)632-1020x37607	632 Michael Grove

customerID	first_name	last_name	email_address	phone_number	address
2000275	Katherine	Anderson	wilsoncrystal@example.com	387-748-4056	24200 Gibson Shore Suite 559
2000276	Ryan	Zamora	xlong@example.org	787.792.1151x21424	9803 George Wells
2000277	Kerri	Davis	denisewilliams@example.com	001-373-800-4811x97189	390 Kevin Drive
2000278	Jessica	Pierce	smithbrian@example.org	+1-600-351-9638x171	32478 Alexis Radial Apt. 999
2000279	Rebecca	Brown	joanochoa@example.net	868-647-8400	25765 Kathryn Mission Apt. 848
2000280	Adam	Gonzalez	tracifischer@example.com	001-621-286-3068x831	84988 Levy Key Suite 759
2000281	Elizabeth	Woods	mjoseph@example.org	803-567-2082x8059	5533 David Stravenue Apt. 051
2000282	Walter	Young	barbara61@example.net	+1-828-312-2130	90608 Silva Fall
2000283	Jessica	Mcmahon	enguyen@example.net	690-382-5710x204	14044 Blair Squares
2000284	Derrick	Davenport	asmith@example.com	(683)937-0462x85671	7575 Mark River
2000285	Denise	Ramos	stephenbradley@example.org	(606)630-7121	7617 Larry Street
2000286	Amanda	Soto	santoslaura@example.org	(205)906-0436x2321	75541 Bobby Ports Suite 549
2000287	Matthew	King	amy30@example.net	001-498-875-6454x0893	9963 Torres Fall
2000288	Matthew	Williams	hdiaz@example.org	841.417.0223x1466	55901 Benson Summit

Rows 0 to 29, Columns 6 to 11

city	state	country	continent	postal_zip_code	gender
Kathrynborough	Massachusetts	Japan	Asia	81587	female
East Catherine	Rhode Island	Japan	Asia	6657	female
Williamshire	Missouri	Japan	Asia	20642	female
Delacruzville	Nevada	Australia	Oceania	21440	male
West Wendy	Colorado	Australia	Oceania	32756	female
Sherryton	Illinois	Japan	Asia	76717	male
East Todd	Iowa	Australia	Oceania	77976	female
Josephburgh	Maryland	Australia	Oceania	2149	male
Kelseyview	Mississippi	Australia	Oceania	85290	female

city	state	country	continent	postal_zip_code	gender
West Deniseland	Connecticut	Japan	Asia	58198	female
Michaelmouth	AK	USA	North America	66393	female
Port Brianton	Washington	Australia	Oceania	32645	male
Willieborough	Kentucky	Australia	Oceania	64820	male
New Michael	Kentucky	Japan	Asia	63656	male
Davieshaven	West Virginia	Japan	Asia	10992	male
Fryeburgh	MS	USA	North America	94447	male
North Charlotte	PW	USA	North America	89124	female
Daniellehaven	Michigan	Japan	Asia	86870	female
East Cherylburgh	UT	USA	North America	39619	female
Moraleston	Idaho	Australia	Oceania	94811	male
South Paige	PA	USA	North America	64255	female
Travisbury	Ohio	Japan	Asia	59506	female
Donaldshire	FM	USA	North America	3067	male
Juliabury	Kansas	Australia	Oceania	95705	female
Grayport	CO	USA	North America	34685	male
Port Michaelborough	Florida	Australia	Oceania	4979	male
New Lauren	Ohio	Japan	Asia	33296	female
Garciaton	KS	USA	North America	63977	female
North Ginaborough	NV	USA	North America	91743	female
New Jamesborough	Idaho	Australia	Oceania	40306	female

Rows 30 to 59, Columns 0 to 5

customerID	first_name	last_name	email_address	phone_number	address
2000289	Jennifer	Ramos	fhunt@example.org	+1-438-770-7979x0794	589 Trujillo Station
2000290	William	Ruiz	william58@example.org	(501)549-6875	3425 Williams Ramp Suite 292
2000291	Dustin	Evans	lisasimpson@example.net	615.732.0839	874 Curtis Vista
2000292	Nathaniel	Sharp	barrkiara@example.com	+1-969-734-3287x5960	3693 Brown Ville Apt. 027
2000293	Jessica	Beck	aevans@example.com	3149322083	275 Perez Turnpike Apt. 342
2000294	Richard	Martinez	cooksara@example.org	895-274-7652	882 Tammy Walk Suite 174
2000295	Carl	Cook	tom16@example.com	001-970-322-5500	5003 Bates Trail
2000296	Thomas	Miller	ericbarton@example.com	807-883-4973	349 Bautista Course
2000297	Nathan	Weaver	jeffrey77@example.net	2344061917	7057 West Junction Suite 274
2000298	Jessica	Golden	zkirk@example.net	(958)204-6182x1563	0411 Jackson Gardens

customerID	first_name	last_name	email_address	phone_number	address
2000299	Gloria	Moore	mholland@example.org	231-280-6373x44149	040 Nelson Throughway
2000222	Megan	Ali	kristenthompson@example.com	001-866-956-2069x3071	796 Morris Way
2000223	Arthur	Mccann	cassidy80@example.net	553-777-9091x6007	62206 Larsen Island Suite 173
2000224	Mary	Dalton	dcarter@example.net	(762)466-3211x19111	223 Blake Mission
2000225	Gary	Lawrence	kylebecker@example.net	395-604-9961x885	019 Waller Trail
2000226	Leslie	Macdonald	wilsondevon@example.org	573-257-7334	09526 Todd Cliff Apt. 602
2000227	Mary	Zuniga	emartinez@example.net	001-651-917-8528x97132	964 Jacob Inlet
2000228	Nicole	Johnson	lewischristopher@example.org	651-356-5102x22222	238 Guerra Prairie
2000229	Jennifer	Holmes	charlescompton@example.com	208.996.6661	75813 King Loop
2000230	Courtney	Knight	parkjeanette@example.net	(377)924-4645x8197	131 Carrie Lane Apt. 570
2000231	Melissa	Lara	nicolepope@example.com	507-852-8306x194	27319 Dennis Trace
2000232	Antonio	Zimmerman	hray@example.org	(571)713-8743x2527	2679 Hart Highway Apt. 979
2000233	Michael	Torres	ggriffin@example.org	(258)333-3397x6318	700 Porter Rapid
2000234	Tracy	Kaufman	parrishedwin@example.org	(816)741-2745	99511 Warren Spurs Suite 971
2000235	Gina	Johnson	loweashley@example.org	001-854-909-7027	98761 Hall Viaduct Suite 154
2000236	Judith	Chandler	juan75@example.net	619-899-1347x7551	50748 Cooper Courts
2000237	Leslie	Bradley	foleyvincent@example.org	+1-971-565-2386x41679	077 Robert Pass Suite 257
2000238	Charles	Martinez	gmartin@example.net	(491)473-5457x40543	271 Brown Stream Apt. 199
2000239	Jennifer	Johnson	markowens@example.org	(388)803-4516	89581 Jessica Canyon Apt. 673
2000240	Brent	Herrera	peter90@example.org	942-947-8385	5202 Jacob Via

Rows 30 to 59, Columns 6 to 11

city	state	country	continent	postal_zip_code	gender
Duffyside	MA	USA	North America	16548	male
New Sarahland	Washington	Japan	Asia	20906	male
Port Reginaburgh	KS	USA	North America	9438	female
Perezhaven	MO	USA	North America	95946	female
Bryantport	Arizona	Japan	Asia	48740	female
Lake Scott	Maine	Australia	Oceania	15719	male

city	state	country	continent	postal_zip_code	gender
North Candice	MO	USA	North America	43738	male
West Michaelshire	MA	USA	North America	21657	male
Jamesville	NY	USA	North America	80363	male
Sharonmouth	PR	USA	North America	96710	female
North Adam	Hawaii	Australia	Oceania	7145	female
East Stephaniestad	OK	USA	North America	52779	female
Brianmouth	West Virginia	Japan	Asia	20733	male
South Michaelberg	Utah	Australia	Oceania	42566	female
Smithland	Alabama	Australia	Oceania	72451	male
West Gregorybury	Wisconsin	Australia	Oceania	26803	female
North Alexandra	Illinois	Australia	Oceania	75725	male
Robinsonstad	Illinois	Australia	Oceania	48237	female
East James	DE	USA	North America	90852	male
West Jamesshire	New York	Japan	Asia	60438	female
Billbury	ND	USA	North America	90058	female
Wardmouth	Montana	Japan	Asia	49821	female
Cookmouth	CA	USA	North America	6893	male
West Sethton	Washington	Australia	Oceania	66516	male
Valeriemouth	MN	USA	North America	9027	female
East George	Mississippi	Japan	Asia	81031	male
Terriland	Connecticut	Japan	Asia	17543	female
West Christophertown	Texas	Australia	Oceania	9433	male
Gonzalezberg	Minnesota	Australia	Oceania	28728	male
Williamsonton	Texas	Australia	Oceania	58628	male

Rows 60 to 89, Columns 0 to 5

customerID	first_name	last_name	email_address	phone_number	address
2000241	Chase	Castillo	crystalmartinez@example.org	(985)593-4391x23005	534 Jonathan Center
2000242	James	Evans	cgarcia@example.com	001-525-662-1156x0500	0863 Rogers Cape
2000243	Robert	Mcgrath	grantrobert@example.net	(203)418-8407x15115	12056 Robert Rapid
2000244	Tina	Garza	cassandra78@example.net	(650)698-3042x083	055 Rowland Light Suite 027
2000245	Juan	Lewis	davidcarr@example.com	+1-398-485-0516	706 Marissa Ports Suite 300
2000246	Jennifer	Alvarez	rushrachel@example.net	(938)896-8714x16027	425 Hudson Run Suite 490

customerID	first_name	last_name	email_address	phone_number	address
2000247	Jesse	Hall	kstewart@example.org	692.941.2259	0685 Edgar Loop Apt. 832
2000248	Chad	Mills	william54@example.net	+1-277-610-0030x390	55032 Gordon Point Apt. 218
2000249	Crystal	Donovan	johnfarmer@example.com	631.724.2438	901 Ballard Square Apt. 759
2000250	Lisa	Cox	taylorbrittany@example.com	611-668-5961x0990	592 Dalton Crossing Suite 097
2000251	Kirk	James	julie08@example.net	730.286.7744	00894 Kemp Dale
2000252	Gregory	Carter	hillkayla@example.net	316-235-7435	5914 Justin Hill Suite 898
2000253	Julie	Boone	smallgregory@example.org	746-953-1518x699	63684 Michael Mill
2000254	Tracy	Farley	theodore30@example.org	001-920-488-9337x581	8495 Berry Bypass Suite 332
2000255	Robert	Mendez	christopher63@example.net	+1-614-510-1452x596	487 Zachary Drive Suite 126
2000256	Janet	Jones	debbiebell@example.net	813.442.5919x05385	808 Taylor Stravenue Suite 535
2000257	Gabriel	Campbell	gutierrezhailey@example.net	(425)572-1099	168 Hunter Harbor Apt. 928
2000258	Charles	Webster	raymond75@example.org	(533)385-3806x697	7469 Rose Fall
2000185	Dawn	Medina	smithchristine@example.com	001-816-421-3013x60329	144 Austin Crossing
2000186	Erin	Brown	awhite@example.net	949-894-0707	94357 Clay Creek
2000187	Sue	Mejia	taylorreyes@example.org	423.823.9822	833 Sarah Points Apt. 974
2000188	Dana	Butler	andreaburns@example.net	(213)541-0600	234 Willis Pike Suite 521
2000189	Scott	Andrews	gary98@example.com	362-263-3989	30415 Turner Knoll Suite 514
2000190	Elaine	Jacobs	xdickerson@example.org	528.787.1534	7909 Johnson Causeway Apt. 890

customerID	first_name	last_name	email_address	phone_number	address
2000191	Rebecca	Archer	shelley23@example.org	694.969.9188x515	66621 James Point Suite 003
2000192	Adrian	Johnson	michealgonzales@example.com	001-292-235-0776x567	25774 Buckley Island
2000193	Sean	Jordan	mosleyjeremiah@example.org	201.853.6511x5916	601 Lowe Isle
2000194	Sara	Blake	brandon57@example.org	504-379-3814x779	32937 Cox Tunnel Suite 030
2000195	Joshua	Peterson	brian71@example.net	(407)367-1106x681	43042 Vincent Drive Suite 749
2000196	Nicholas	Hobbs	mcdowelljohn@example.com	001-285-455-1732	7383 Calderon Trail

Rows 60 to 89, Columns 6 to 11

city	state	country	continent	postal_zip_code	gender
Smithmouth	Rhode Island	Japan	Asia	82996	male
Samuelview	Hawaii	Australia	Oceania	55946	male
Harrisfort	Maryland	Australia	Oceania	44248	male
North Jasonville	Vermont	Australia	Oceania	19226	male
South Lindatown	TN	USA	North America	92443	male
East Heatherbury	Arkansas	Japan	Asia	45989	female
Jamesview	AK	USA	North America	14944	male
West Maria	Arizona	Australia	Oceania	52103	male
East Hollyborough	Maine	Australia	Oceania	55574	female
Rebeccamouth	Texas	Japan	Asia	5607	female
East Beverlybury	Colorado	Japan	Asia	42283	male
Neilberg	New Mexico	Japan	Asia	6651	male
Mollyborough	Texas	Japan	Asia	4758	male
Port Robin	IL	USA	North America	78903	male
West Leah	MP	USA	North America	93462	male
Kingtown	Illinois	Japan	Asia	45002	male
Craigmouth	New Jersey	Japan	Asia	15896	male
North Jacob	Mississippi	Australia	Oceania	87182	male
Port Jenniferton	Kansas	Australia	Oceania	65361	female
Jeffreyville	Nebraska	Japan	Asia	23853	male
Barrberg	NE	USA	North America	73571	female
Elliottshire	New Jersey	Japan	Asia	35429	male
Robinsonberg	Georgia	Australia	Oceania	22662	male
North Michellemouth	Alabama	Japan	Asia	19360	male

city	state	country	continent	postal_zip_code	gender
Port Daniel	New York	Japan	Asia	25317	female
Lake Glenda	Missouri	Australia	Oceania	72307	female
New Jamesstad	Iowa	Japan	Asia	96139	female
East Amanda	TX	USA	North America	90828	male
West Denniston	Nevada	Australia	Oceania	96320	female
Mariefort	Washington	Australia	Oceania	68352	male

Rows 90 to 119, Columns 0 to 5

customerID	first_name	last_name	email_address	phone_number	address
2000197	Alexander	Diaz	brianbartlett@example.org	+1-893-876-1543x750	3977 Wallace Trace
2000198	Andrew	Jackson	hreed@example.org	(762)691-5934	088 Howell Way Apt. 637
2000199	Briana	Campbell	theodore94@example.com	001-284-731-2302x4594	125 Robert Prairie Suite 565
2000200	Jessica	Kelley	williamdavis@example.net	(883)976-3731x109	2780 Jeffrey Knolls Suite 493
2000201	Mark	Mata	ugonzalez@example.com	+1-479-285-3357x9170	7238 Parker Viaduct Apt. 158
2000202	Joshua	Davis	esnow@example.net	(946)977-8975	51942 Jesse Radial
2000203	Jenna	Gilbert	cbates@example.com	646-209-2215x522	043 Christian Springs Suite 005
2000204	Robert	Massey	loriestrada@example.net	001-708-383-2908x28900	186 Kramer Ranch Apt. 835
2000205	Jeffery	Green	donna29@example.org	254-984-2052	776 Zachary Hills
2000206	Charles	Cook	thomasrachel@example.com	786.300.4605x012	713 Jennifer Overpass Suite 698
2000207	Betty	Casey	victorparsons@example.org	001-494-302-8652x19122	7629 Jacobs Island Apt. 589
2000208	John	Ponce	kgreen@example.com	001-576-206-7718	640 Jonathan Heights
2000209	Cameron	Rice	emily18@example.com	512.423.1043x25744	98269 Jay Prairie
2000210	Tom	Lambert	robertfischer@example.org	+1-569-333-7100x8329	631 Kimberly Spur
2000211	Tina	Erickson	uedwards@example.org	+1-508-670-6974x01967	7938 Kendra Turnpike

customerID	first_name	last_name	email_address	phone_number	address
2000212	Lawrence	Turner	dhensley@example.com	243-905-6469x96028	234 Tiffany Mountain Apt. 808
2000213	Kelly	Ball	evansmike@example.net	+1-445-832-6303x47297	7465 Yates Burgs Apt. 023
2000214	Benjamin	Dorsey	sandrablevins@example.com	+1-722-652-2895x0183	070 Hannah Row Suite 870
2000215	Nicole	Zuniga	jessicalawrence@example.com	588-380-1347x250	310 Wood Pines Apt. 776
2000216	Rebecca	Mitchell	jkrueger@example.org	+1-568-388-9570x3994	32353 Louis Landing Apt. 060
2000217	Mia	Reed	wmills@example.org	(269)497-8688x78549	406 Snyder Park Suite 057
2000218	Tiffany	Garcia	adkinsfrederick@example.com	(958)990-3188x025	944 Williams Prairie
2000219	Joshua	Griffin	samuelwilliams@example.org	457-674-5874	58640 Martin Tunnel Suite 831
2000220	Christine	Rivera	lanemegan@example.org	(727)286-8565	72537 Durham Shores Suite 208
2000221	Kaitlyn	Martinez	cassandrashaw@example.org	545.967.7847	566 Thomas Springs
2000111	Kevin	Bowen	adam15@example.com	+1-587-890-1539	59876 Hammond Motorway
2000112	Lorraine	James	floreskayla@example.net	466.557.6091x1511	0668 Joshua Square Apt. 830
2000113	Brian	Miller	smithderek@example.net	507.292.1337	9730 Brian Shoals
2000114	Benjamin	Edwards	mknapp@example.org	001-931-908-7984	3192 David Tunnel Apt. 471
2000115	Barry	Taylor	george89@example.org	(237)532-2214x8562	522 Hopkins Walk Suite 923

Rows 90 to 119, Columns 6 to 11

city	state	country	continent	postal_zip_code	gender
South Lindaton	Virginia	Japan	Asia	34215	female
Villegasville	HI	USA	North America	5562	male

city	state	country	continent	postal_zip_code	gender
Galvanborough	Missouri	Japan	Asia	52649	male
Kristinville	Wisconsin	Australia	Oceania	63903	male
East Thomasberg	MS	USA	North America	56155	female
East Alexiston	Kansas	Australia	Oceania	98394	male
Melissamouth	KY	USA	North America	71047	male
Johnfurt	NH	USA	North America	46416	female
Joshuamouth	Iowa	Australia	Oceania	76708	female
Lake Jamesport	ME	USA	North America	92670	female
Lake Williamfort	Connecticut	Australia	Oceania	9149	male
Anthonyview	MS	USA	North America	66970	female
Jacobmouth	PA	USA	North America	76277	female
New Jose	Texas	Australia	Oceania	64407	male
South Frankfurt	Massachusetts	Japan	Asia	57211	male
Wendyshire	AZ	USA	North America	11904	male
Davidbury	NE	USA	North America	55738	female
Lake Dustin	OK	USA	North America	39970	female
South Lindseymouth	California	Japan	Asia	12426	female
North Ryan	Nevada	Japan	Asia	60932	female
West Sarahton	South Dakota	Australia	Oceania	52410	female
Floydborough	Alabama	Japan	Asia	57357	female
Andreafurt	Nevada	Japan	Asia	1661	female
Taylorshire	Arizona	Australia	Oceania	48488	female
Port Ianshire	MD	USA	North America	47795	male
New Davidview	Connecticut	Australia	Oceania	48820	female
North Joshuachester	Kentucky	Australia	Oceania	36759	female
Ericksonton	AR	USA	North America	88982	male
New Amber	Washington	Japan	Asia	55805	female
Brownport	Michigan	Australia	Oceania	36556	female

Rows 120 to 149, Columns 0 to 5

customerID	first_name	last_name	email_address	phone_number	address
2000116	Lindsey	Garcia	xgreen@example.org	970.829.6757x12935	027 Hartman Row Suite 437
2000117	Thomas	Gibson	holmeselizabeth@example.com	(876)676-7766x76584	030 Lopez Inlet Apt. 806
2000118	Todd	Wilson	loganbarnes@example.com	001-575-414-6958x1843	2698 Leblanc Junctions
2000119	Tiffany	Bowman	elizabeth04@example.com	977.496.0897	7100 Webster Meadow

customerID	first_name	last_name	email_address	phone_number	address
2000120	Ashley	Turner	xzhang@example.com	001-725-844-1381x01435	484 Sara Harbor Apt. 845
2000121	Tamara	Hughes	robertdixon@example.org	001-462-823-7921x8932	73105 Erica Crest
2000122	Tina	Cruz	cfoster@example.net	(258)686-4697x92986	22045 Brandy Street
2000123	Richard	Mullins	schmidtjames@example.org	9216374288	0321 Ryan Walks Apt. 871
2000124	Natalie	Garcia	chadcollins@example.org	389-887-3173	57423 Linda Fords
2000125	Nicole	Williams	elijah33@example.com	+1-425-594-6236x460	53948 Andrea Valleys
2000126	Brendan	Garcia	jbrooks@example.com	(760)817-0212	86896 Spencer Villages Suite 995
2000127	Ashley	Chavez	margaret27@example.org	+1-885-244-1914	615 Norris Fort
2000128	David	Ritter	webermelanie@example.org	665.465.2736	88571 Benson Orchard
2000129	Brianna	Martin	slin@example.org	+1-468-952-2963x80194	0850 Jessica Key Apt. 126
2000130	Lori	Smith	anthony64@example.net	871.612.8119x06680	93716 Snyder Estates Suite 462
2000131	Kimberly	Simmons	jacqueline34@example.org	+1-710-513-2701	36362 Michael Park
2000132	Shelly	Johnson	roberta21@example.com	+1-517-351-9600x7418	7231 Nash Isle
2000133	Leslie	Miller	qroberts@example.com	822-424-6013x91719	36929 Jennifer Mount Apt. 202
2000134	Charles	Wong	mcrawford@example.org	7222934327	995 Shannon Overpass
2000135	Brian	Mercado	hmcintyre@example.com	001-302-637-9966x12729	935 Randy Ways
2000136	Thomas	Manning	tvaughan@example.org	462.527.2954	1892 James Row Suite 589

customerID	first_name	last_name	email_address	phone_number	address
2000137	Mikayla	Ho	jaredpace@example.org	001-884-801-8046x457	700 Michael Points Suite 466
2000138	Shawna	Banks	devans@example.net	276-715-3718x764	0319 Shannon Junction
2000139	William	Wagner	brownbrian@example.net	(540)622-7482	37384 Roberts Gateway Suite 891
2000140	Eric	Stephenson	christopher75@example.net	001-924-973-5296x7924	6280 Lynn Drive
2000141	Danny	Roberts	bestbailey@example.org	4522474105	52830 Ronnie Harbors Suite 112
2000142	Harry	Allen	mnewman@example.com	+1-401-302-5000	38880 Joseph Lights
2000143	Amy	Christian	catherine87@example.com	+1-275-385-4375x940	5591 James Burg
2000144	Wesley	Jordan	ryan71@example.com	503.782.1412x11999	18023 Maria Ridges Apt. 078
2000145	Ann	Bullock	jeremyfischer@example.net	+1-234-882-8857x4374	692 Nelson Alley Suite 982

Rows 120 to 149, Columns 6 to 11

city	state	country	continent	postal_zip_code	gender
Antonioshire	NM	USA	North America	59560	female
Palmerview	DC	USA	North America	60906	female
Halltown	MN	USA	North America	22699	female
Zhangburgh	MD	USA	North America	53520	male
Danielmouth	Kentucky	Japan	Asia	97355	female
Smithfort	AL	USA	North America	32925	female
South Amy	MT	USA	North America	28187	male
New Emmaview	Tennessee	Australia	Oceania	2779	male
Rodriguezside	Vermont	Japan	Asia	46497	female
Johnsonstad	Louisiana	Japan	Asia	60369	male
Michaelburgh	PW	USA	North America	73692	male
Philiptown	California	Japan	Asia	45793	male
Elizabethshire	Oregon	Australia	Oceania	55264	female
Beverlyshire	Arkansas	Japan	Asia	56265	male
New Michaelfurt	Oregon	Australia	Oceania	13827	female
New Timothyside	AL	USA	North America	32746	female
New Stanley	South Carolina	Australia	Oceania	41933	male

city	state	country	continent	postal_zip_code	gender
Port Johnburgh	GU	USA	North America	58509	female
Houstonside	New Jersey	Japan	Asia	60812	female
East Joseph	South Dakota	Australia	Oceania	98071	male
Javierfort	PW	USA	North America	1123	male
Tamichester	Delaware	Australia	Oceania	27401	female
Port Laurenview	Arizona	Japan	Asia	34514	female
Fosterstad	FM	USA	North America	5772	female
New Richardmouth	MD	USA	North America	23053	male
Port Cherylhaven	Illinois	Australia	Oceania	35139	male
Lake Jeffreyhaven	TX	USA	North America	46102	female
Castroburgh	OH	USA	North America	36660	female
East Audreystad	California	Australia	Oceania	49459	female
Juanland	New Mexico	Australia	Oceania	46389	female

Rows 150 to 179, Columns 0 to 5

customerID	first_name	last_name	email_address	phone_number	address
2000146	Dustin	Meyers	maryschmitt@example.net	(813)928-0087x664	0262 Berry Shores Apt. 038
2000147	Kimberly	Oconnell	nwelch@example.com	001-381-697-5054	969 Clark Shoal Apt. 166
2000037	Cynthia	Vega	mitchellcolleen@example.net	(509)293-2155x5876	528 Allen Turnpike
2000038	Gregory	Flores	plucas@example.org	001-779-903-6659x127	499 Scott Isle
2000039	Kyle	Jennings	wallacechristopher@example.net	+1-963-751-6331x266	0667 Bryant Throughway
2000040	Jesse	Mcmillan	ashleyguerra@example.net	(739)460-0995	698 Kevin Plains
2000041	Kathleen	Rogers	walkerjoshua@example.org	001-717-261-5326x436	14589 Martinez Hill
2000042	Richard	Smith	wdavidson@example.net	797.676.9570x485	071 Crawford Ridges Apt. 309
2000043	Kelly	Adams	denisewilliams@example.net	001-587-353-5500x53146	13185 Steven Dam Apt. 117
2000044	James	Snow	willie41@example.com	(673)529-2841	24635 Greene Groves Suite 597
2000045	Heidi	Conrad	lawrencedavis@example.org	(592)492-1197x53437	67752 Tate Circles Suite 879

customerID	first_name	last_name	email_address	phone_number	address
2000046	Darren	Howard	jonescarolyn@example.net	753-404-9713x9416	142 Abbott Ridges Apt. 691
2000047	Andrea	Sanchez	johnjackson@example.net	(388)500-2993	061 Stephens Stream
2000048	Elizabeth	Watkins	williamsruth@example.net	562-451-7585	55000 Jacobson Port Suite 319
2000049	Melissa	Peters	ebrooks@example.org	422-765-5148x511	52584 Gary Burg
2000050	Justin	Rowe	michael64@example.com	910-968-7522x7426	52350 Bailey Stravenue
2000051	Michael	Crawford	lauriejackson@example.org	+1-562-578-5384x4238	0727 Marsh Ranch
2000052	Brian	Thompson	taylor74@example.org	+1-455-913-4582x52112	389 Copeland Coves
2000053	Phillip	Lynch	hoconnor@example.net	645.456.8157x3024	1097 Wong Tunnel Apt. 984
2000054	Mark	Spencer	hklein@example.org	001-902-420-4665x213	10684 Smith Unions
2000055	Jessica	Armstrong	ujohnson@example.com	(784)453-3248x50625	77892 Ryan Cape
2000056	Oscar	Goodwin	qmaddox@example.net	3649557328	04865 Mary Rapids Apt. 660
2000057	Kelly	Davis	timothy20@example.net	963-849-8105x74952	9438 Carr Squares Suite 830
2000058	James	Sanchez	conniedoyle@example.com	292.881.0037x20775	50611 Cook Drive
2000059	Dakota	Robertson	stacymeyer@example.org	757.795.7169x37003	341 Carlson Points
2000060	Sean	Ramirez	kellysullivan@example.com	220-729-3501	2419 Turner Mountains Apt. 293
2000061	Darin	Campbell	misty73@example.com	425-963-5073	342 Jacqueline Isle Apt. 684
2000062	Megan	Sherman	fgarcia@example.org	613.251.3520x363	01746 Sanchez Trafficway Suite 562

customerID	first_name	last_name	email_address	phone_number	address
2000063	Shane	Green	reneevillanueva@example.net	438-377-0234x50378	91313 Hunter Center
2000064	Amanda	Murray	leenatalie@example.net	+1-881-205-2480x436	4221 Miranda Avenue

Rows 150 to 179, Columns 6 to 11

city	state	country	continent	postal_zip_code	gender
East Carla	Nevada	Japan	Asia	98591	female
Hansenberg	Indiana	Australia	Oceania	21955	male
Justinshire	Iowa	Japan	Asia	36533	male
West Malikland	ND	USA	North America	38548	female
New Jerryview	AS	USA	North America	73206	male
East Antoniomouth	Minnesota	Japan	Asia	73722	male
Lake Donnaberg	Arkansas	Japan	Asia	90376	female
Edwardton	South Dakota	Australia	Oceania	7775	male
New Eric	NH	USA	North America	97787	female
North Howardborough	HI	USA	North America	51371	male
Port Scottton	MD	USA	North America	92126	male
Banksmouth	Virginia	Australia	Oceania	52880	male
South Kevinbury	WI	USA	North America	65582	male
New Linda	New York	Japan	Asia	21521	male
West Erinburgh	Ohio	Australia	Oceania	35966	female
New Destiny	OH	USA	North America	85598	female
Loriside	California	Japan	Asia	63070	female
North Nicole	Illinois	Japan	Asia	7174	female
Port Angela	Wisconsin	Australia	Oceania	8228	male
South Mariatown	Utah	Japan	Asia	90509	male
Lake Patrick	NM	USA	North America	96264	male
Kellybury	Georgia	Japan	Asia	24054	male
Garcialand	New Hampshire	Japan	Asia	40749	male
Jenniferstad	Indiana	Australia	Oceania	75896	female
East James	PR	USA	North America	28094	male
Jeffreyshire	Nevada	Australia	Oceania	13085	female
Davisburgh	Maine	Japan	Asia	38855	male
Payneberg	Ohio	Japan	Asia	85246	male
West Matthewview	Wyoming	Japan	Asia	40178	female
Brookemouth	New Jersey	Japan	Asia	13581	male

Rows 180 to 199, Columns 0 to 5

customerID	first_name	last_name	email_address	phone_number	address

customerID	first_name	last_name	email_address	phone_number	address
2000065	Matthew	Acosta	washingtonshannon@example.org	823-703-6329x60550	5547 Powers Station Apt. 732
2000066	Nancy	Clarke	tiffanycurtis@example.com	001-447-301-0540	03790 Wallace Corners Suite 508
2000067	Christopher	Williams	cgraham@example.net	392.267.5287x66973	4768 Sharp Forest
2000068	Jesse	Clark	watsonrobert@example.com	472.293.1258	70165 Martin Junction Apt. 784
2000069	Michelle	Chen	ashleyanderson@example.com	+1-928-811-1758x410	160 Moore Turnpike Apt. 957
2000070	Martin	Marshall	charles59@example.net	001-824-258-2310x6495	00367 Stevens Stravenue
2000071	Mary	Mason	reevesjoshua@example.com	543-532-4156	9259 James Lock
2000072	Robert	Santiago	qruiz@example.org	+1-829-436-4296	52800 Thompson Mountain
2000073	Tracy	Higgins	scotttammy@example.com	577.853.2129	66366 Daniel Expressway Suite 642
2000148	Douglas	Barnes	lopezelizabeth@example.com	756.464.3566x5152	11032 Gregory Spurs Apt. 312
2000149	Makayla	Walsh	moorepatrick@example.org	(337)988-3880	48044 Martinez Loaf Suite 632
2000150	Bryan	Jackson	christinacervantes@example.net	+1-865-762-9132	5520 Heather Grove Apt. 933
2000151	Eric	Kennedy	ysloan@example.com	+1-663-946-3962x092	7091 Nicole Wells Apt. 343
2000152	Jeffery	Whitney	erios@example.net	4148110817	658 Monica Drives
2000153	Shelby	Pratt	ojohnston@example.com	001-212-328-1233	85997 Evan Cliffs

customerID	first_name	last_name	email_address	phone_number	address
2000154	Meredith	Wilson	johnny60@example.org	(353)370-3255x134	23020 Davis Ferry Suite 249
2000155	Jessica	Green	roachadam@example.com	437.418.3006	24071 Douglas Spurs
2000156	Megan	Horton	danielcoleman@example.com	001-719-961-9988x239	074 Kristin Lake Apt. 115
2000157	Laura	Morris	wgray@example.net	6592313215	50431 Obrien Highway Suite 109
2000158	Diana	Moore	mariomcgrath@example.net	+1-383-810-6611x189	15711 Mccarthy Meadows Apt. 925

Rows 180 to 199, Columns 6 to 11

city	state	country	continent	postal_zip_code	gender
Williamsstad	PW	USA	North America	84340	female
East Donnaville	HI	USA	North America	9351	female
Lake Kaitlin	Utah	Australia	Oceania	43022	male
Port Moniqueborough	Kentucky	Australia	Oceania	12116	male
Glenville	IN	USA	North America	38254	female
Jonesburgh	Hawaii	Australia	Oceania	18934	male
Traviston	Arkansas	Australia	Oceania	14321	female
New James	OR	USA	North America	55760	female
Jamesberg	MI	USA	North America	6181	male
Griffinberg	Minnesota	Japan	Asia	65222	male
Rachelborough	IN	USA	North America	30565	male
Hamptonfurt	New Mexico	Australia	Oceania	79722	female
Kaylachester	DC	USA	North America	47330	female
Anthonyport	Alaska	Australia	Oceania	61816	male
Kyleshire	Pennsylvania	Australia	Oceania	88834	male
Tranmouth	Mississippi	Japan	Asia	63586	male
West Claytonshire	Georgia	Japan	Asia	27240	male
New Jessica	OR	USA	North America	67334	female
Port Chad	Missouri	Japan	Asia	69630	male
Tommystad	Colorado	Japan	Asia	12947	female