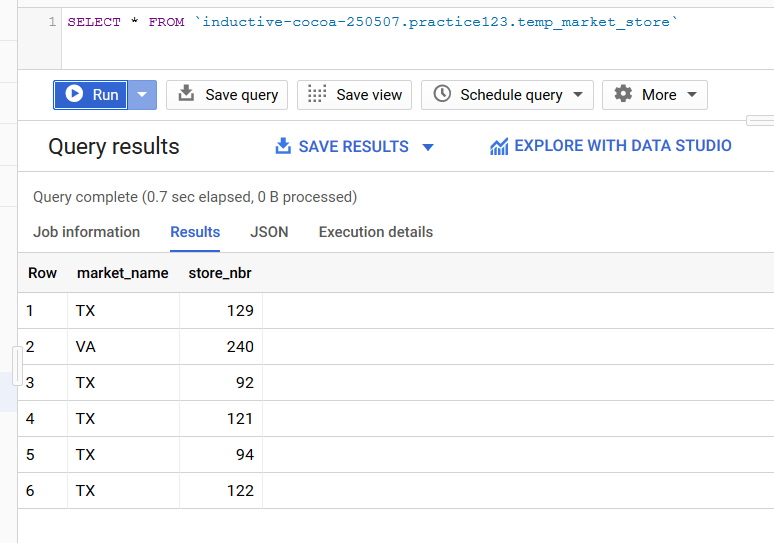
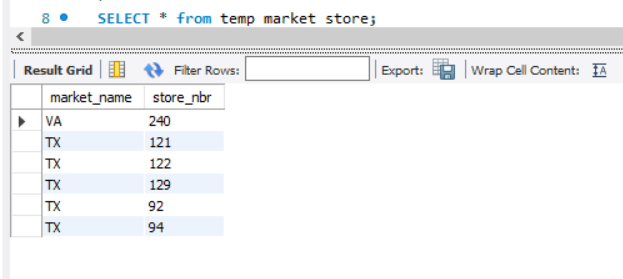
# **POC on Data Ingestion from mysql to BQ using Python**

* Generate google service account key
  + IAM & Admin > Service account > create\_Service\_account
  + Once created then create key , download and save It to the project folder on local machine – google\_key.json
* Run the code in pycharm environment after installing the packages.
* Data validation : Count and schema
  + BigQuery Output:
  + Mysql Input :

**NOTE : The table data in mysql remains intact. Also , if one uses preview in BQ to see that you won’t see. Go to console and fire the query.**

* **CODE**
* **import** MySQLdb  
  **from** google.cloud **import** bigquery  
  **import** mysql.connector  
  **import** logging  
  **import** os  
  **from** MySQLdb.converters **import** conversions  
  **import** click  
  **import** MySQLdb.cursors  
  **from** google.cloud.exceptions **import** ServiceUnavailable  
  **import** sys  
    
  bqTypeDict = {**'int'**: **'INTEGER'**,  
   **'varchar'**: **'STRING'**,  
   **'double'**: **'FLOAT'**,  
   **'tinyint'**: **'INTEGER'**,  
   **'decimal'**: **'FLOAT'**,  
   **'text'**: **'STRING'**,  
   **'smallint'**: **'INTEGER'**,  
   **'char'**: **'STRING'**,  
   **'bigint'**: **'INTEGER'**,  
   **'float'**: **'FLOAT'**,  
   **'longtext'**: **'STRING'**,  
   **'datetime'**: **'TIMESTAMP'** }  
    
    
  **def** conv\_date\_to\_timestamp(str\_date):  
   **import** time  
   **import** datetime  
    
   date\_time = MySQLdb.times.DateTime\_or\_None(str\_date)  
   unix\_timestamp = (date\_time - datetime.datetime(1970, 1, 1)).total\_seconds()  
    
   **return** unix\_timestamp  
    
    
  **def** Connect(host, database, user, password):  
   **return** mysql.connector.connect(host=**'tempus-qa.hashmapinc.com'**,  
   port=**'30656'**,  
   database=**'recommendation\_spark'**,  
   user=**'root'**,  
   password=**'docker'**)  
    
    
  **def** BuildSchema(host, database, user, password, table):  
   logging.debug(**'build schema for table %s in database %s'** % (table, database))  
   conn = Connect(host, database, user, password)  
   cursor = conn.cursor()  
   cursor.execute(**"DESCRIBE %s;"** % table)  
    
   tableDecorator = cursor.fetchall()  
   schema = []  
    
   **for** col **in** tableDecorator:  
   colType = col[1].split(**"("**)[0]  
   **if** colType **not in** bqTypeDict:  
   logging.warning(**"Unknown type detected, using string: %s"**, str(col[1]))  
    
   field\_mode = **"NULLABLE" if** col[2] == **"YES" else "REQUIRED"** field = bigquery.SchemaField(col[0], bqTypeDict.get(colType, **"STRING"**), mode=field\_mode)  
    
   schema.append(field)  
    
   **return** tuple(schema)  
    
    
  **def** bq\_load(table, data, max\_retries=5):  
   logging.info(**"Sending request"**)  
   uploaded\_successfully = **False** num\_tries = 0  
    
   **while not** uploaded\_successfully **and** num\_tries < max\_retries:  
   **try**:  
   insertResponse = table.insert\_data(data)  
    
   **for** row **in** insertResponse:  
   **if 'errors' in** row:  
   logging.error(**'not able to upload data: %s'**, row[**'errors'**])  
    
   uploaded\_successfully = **True  
   except** ServiceUnavailable **as** e:  
   num\_tries += 1  
   logging.error(**'insert failed with exception trying again retry %d'**, num\_tries)  
   **except** Exception **as** e:  
   num\_tries += 1  
   logging.error(**'not able to upload data: %s'**, str(e))  
    
    
  @click.command()  
  @click.option(**'-h'**, **'--host'**, default=**'tempus-qa.hashmapinc.com'**, help=**'MySQL hostname'**)  
  @click.option(**'-d'**, **'--database'**, required=**True**, help=**'MySQL database'**)  
  @click.option(**'-u'**, **'--user'**, default=**'root'**, help=**'MySQL user'**)  
  @click.option(**'-p'**, **'--password'**, default=**'docker'**, help=**'MySQL password'**)  
  @click.option(**'-t'**, **'--table'**, required=**True**, help=**'MySQL table'**)  
  @click.option(**'-i'**, **'--projectid'**, required=**True**, help=**'Google BigQuery Project ID'**)  
  @click.option(**'-n'**, **'--dataset'**, required=**True**, help=**'Google BigQuery Dataset name'**)  
  @click.option(**'-l'**, **'--limit'**, default=0, help=**'max num of rows to load'**)  
  @click.option(**'-s'**, **'--batch\_size'**, default=1000, help=**'max num of rows to load'**)  
  @click.option(**'-k'**, **'--key'**, default=**'key.json'**,help=**'Location of google service account key (relative to current working dir)'**)  
  @click.option(**'-v'**, **'--verbose'**, default=0, count=**True**, help=**'verbose'**)  
  **def** SQLToBQBatch(host, database, user, password, table, projectid, dataset, limit, batch\_size, key, verbose):  
   *# set to max verbose level* verbose = verbose **if** verbose < 3 **else** 3  
   loglevel = logging.ERROR - (10 \* verbose)  
    
   logging.basicConfig(level=loglevel)  
    
   logging.info(**"Starting SQLToBQBatch. Got: Table: %s, Limit: %i"**, table, limit)  
   *## set env key to authenticate application* os.environ[**'GOOGLE\_APPLICATION\_CREDENTIALS'**] = os.path.join(os.getcwd(), key)  
   print(**'file found'**)  
   *# Instantiates a client* bigquery\_client = bigquery.Client()  
   print(**'Project id created'**)  
    
   **try**:  
    
   bq\_dataset = bigquery\_client.dataset(dataset)  
   bq\_dataset.create()  
   logging.info(**"Added Dataset"**)  
   **except** Exception **as** e:  
   **if** (**"Already Exists: " in** str(e)):  
   logging.info(**"Dataset already exists"**)  
   **else**:  
   logging.error(**"Error creating dataset: %s Error"**, str(e))  
    
    
    
   bq\_table = bq\_dataset.table(table)  
   bq\_table.schema = BuildSchema(host, database, user, password, table)  
   print(**'Creating schema using build schema'**)  
   bq\_table.create()  
   logging.info(**"Added Table %s"**, table)  
    
   conn = Connect(host, database, user, password)  
   cursor = conn.cursor()  
    
   logging.info(**"Starting load loop"**)  
   cursor.execute(**"SELECT \* FROM %s"** % (table))  
    
   cur\_batch = []  
   count = 0  
    
   **for** row **in** cursor:  
   count += 1  
    
   **if** limit != 0 **and** count >= limit:  
   logging.info(**"limit of %d rows reached"**, limit)  
   **break** cur\_batch.append(row)  
    
   **if** count % batch\_size == 0 **and** count != 0:  
   bq\_load(bq\_table, cur\_batch)  
    
   cur\_batch = []  
   logging.info(**"processed %i rows"**, count)  
    
   *# send last elements* bq\_load(bq\_table, cur\_batch)  
   logging.info(**"Finished (%i total)"**, count)  
   print(**"table created"**)  
    
    
  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  
   *# run the command* SQLToBQBatch()
* ***Command to run the file :*** *python mysql\_to\_bq.py -d 'recommendation\_spark' -t temp\_market\_store -i inductive-cocoa-250507 -n practice123 -k key.json*
* **Key.json file :** We need to generate it. This key is for my account.

{  
 **"type"**: **"service\_account"**,  
 **"project\_id"**: **"big-involution-250507"**,  
 **"private\_key\_id"**: **"4750e43183aa473218cc1b9a696d3c9311c66cdb"**,  
 **"private\_key"**: **"-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCVZb7hY6JWsTv2\n88KlHR3VOLfyxYUGgKG7WScfv+9bsLH7dY7c3WOuSU7fp4B4TAnIumo3ME6VVQzp\nRfZHx4ASYArvecXDCShkfpyb6/uDBqvCH197t2N+Dy8S4NaP43vzvC+mljBUuQWE\n25srMDoz9PdlAn/xOtYAc3l3Wq36Ota4ed0L+FS3DXF3MaRY9twc0AtZTnIov86b\ntcRyCS6FC6vJr58xYfSwXoaY44PkivCWEGvdWKaz48BzJmO4wgj/TdrDQgpBrwdT\njVtZy9g1gzSdVFhlC3Vsjc+q6MPBZKKeGUWcPM6jott394+v4vJdP6Du2/YGg5aE\nKbM0yJXnAgMBAAECggEAIcX0DmDnQKDldPDrNkdsSJj/YKcNaQhoR4GiihdqRXie\n68FKMBzuXWaeKYpkzQ0ngxS9UUee5DxrZErYH6b2iXDj1/pX3jzyQDGtNsR0wcye\nxI4yGDY/jC5jjml8p9+d1F5dCVRXjbecC0pX/D31UmyVBuU3RjHuUgDI21YJEZGm\nL+gktXcAxCdVC1SRtNItFtRrzVNDYQt9FmwnWGkAc4Z+6Avl3o8jaBW9vL8l2nvV\nm1Mn6li1AZvfUcqMwJDjQxF4g6Q+XZiuLoNXpeT4Rauv7DOHWSona1nt8H+i/+pf\nbjPDNnIVJY6K6uveU0pzMasTFxesi+pXHlYiUckgmQKBgQDED4y3E77+xTAcCMcj\n0sW8qBeeL9alqMIATwURzEsXSi0i9OVwUhFmVd2dgP9PiJTIuozlYsv/1LDGoIc/\n23mS+wfmE5NEOAy82MiISEg4iYkPiGGjh8X4pUbhKT8StYtEhDB8B7RoT8W6SqmT\nlViGLa/QP77u0uHyt5plQFLyfwKBgQDDEiamyMEcl6glGTx6b2cUJ73MR5xDV0rl\nDJa/nUdDmjvuBHmhAI9J7sPjRTvpej+tWF5llD3RapwcxpbEf5wuhCBuvp26P1HT\nVgzVMI3gRITEACUOpwGNg2PEADtP5KCkgz1wkBQMcoosLze5k3SvhVKMG8/xdoD9\nfgnqPyZYmQKBgQCMf2gjOe9AuyYCsPej4EL7MqnaHgX/qLN9bSMA4hldkzZNOpnA\np874OsgZOnu7DvEv9x9IskfmCRC1Bgxl4m14upOqeJSr6k6WdfMeVk4K1eR2URFH\nEuD3kdj9OYWNbrNiYigseYj6x0+2dBKZRAiv/sViwoucsVphiNkHfdTzGQKBgQDB\n9Ggt6OnpjfkHegIiEmZBKgqS9JiwdDjpaQzyVGUpGE+CJzzwEmhmMFrgvFQHnx91\nMaoaAb09XWz/R41cry0RKxo8Qpb9OjwXjzWPO4m1dSs03pf565+lnwAxP7G5jC8V\no9AQ/ZCRl8k+iWw0viKE6mDbIkQtZ5B1ez5MhKGR8QKBgAaTBVvg5lXOfLxexc24\nLhwTiLwnU1MuFktVhpRXcGzqJfzu/v+fhbIgffVuVqi6Ww0/yXx+KIxTGUbR8pvZ\nSWj+ZyMzjXY/kCSjkcMbFfx1f38qItQYU6+bg75uF2viHa0LTpU+C3IzBQP5fYjU\n+EEzBo1s9wrMpv9uVnQX1uCS\n-----END PRIVATE KEY-----\n"**,  
 **"client\_email"**: **"mysqltobq@big-involution-250507.iam.gserviceaccount.com"**,  
 **"client\_id"**: **"117908162608507448396"**,  
 **"auth\_uri"**: **"https://accounts.google.com/o/oauth2/auth"**,  
 **"token\_uri"**: **"https://oauth2.googleapis.com/token"**,  
 **"auth\_provider\_x509\_cert\_url"**: **"https://www.googleapis.com/oauth2/v1/certs"**,  
 **"client\_x509\_cert\_url"**: **"https://www.googleapis.com/robot/v1/metadata/x509/mysqltobq%40big-involution-250507.iam.gserviceaccount.com"**}