DATABASE PROJECT

Akanksha

12/03/2020

Section 1

The plan

We need a database to store daily stock data for tickers, generated as csv files. Each csv file contains an individual ticker's data for a particular timeframe.

The following calculations give an insight on the size of data we are dealing with.

There are approx. 2000 tickers, with each ticker's data-points from year 2008 to present - for all working days, which count to 261-262 days per year

For each ticker, there are up-to 390 data-points per day, since data is being collected from 930am until 4pm in 1 min intervals

By 31 Dec 2020:

If each ticker has data-points for every minute of every working day from 930am-4pm in the years 2008-2020:

For 1 ticker: 390 * 262 * 13 = 1,328,340 rows

For 2000 tickers: 1,328,340 * 2000 = 2,656,680,000 rows

Keeping above estimations in mind, the database is designed to distribute the tickers into 10 alphabetical groups, such that there are an approximately equal number of tickers in each group. Secondly, each ticker's data has been divided into 13 groups, year-wise [2008-2020].

y2008 - 10 tables, y2009 - 10 tables, y2010 - 10 tables, **up-to** y2020 - 10 tables

This makes 130 tables in total.

Each table will have a particular year's data for [2000 tickers / 10 bins] = 200 tickers

In each table:

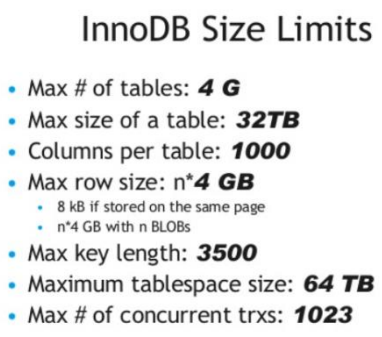
For 1 ticker: 390 (data-points/day) * 262 (days) = 102,180 rows

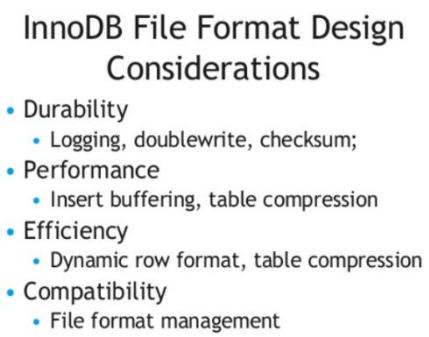
For 10 tickers: 390 * 262 * 10 rows = 1,021,800 rows

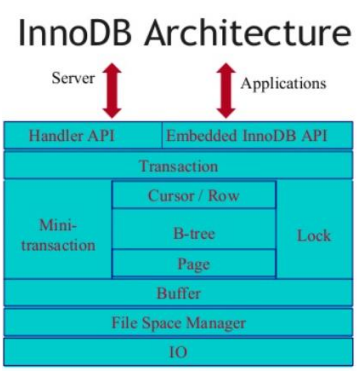
For (the maximum) 200 tickers: 390 * 262 * 2000 = 20,436,000 rows

Thus, each table has maximum 20,436,000 rows

Consequently, for 130 tables: 20436000 * 130 = 2,656,680,000 rows







https://www.slideshare.net/zhaolinjnu/inno-db-internals-innodb-file-formats-and-source-code-structure

Section 2

The ER Diagram. The Schema and Table Structures

Y2008

DTP_1	
id	int
ds_id	char(15)
open	float
high	float
low	float
close	float
volume	decimal

DTP_2	
id	---------
ds_id	----------
open	---------
high	---------
low	---------
close	----------
volume	decimal

DTP_10	
---------	-------
---------	-------
---------	-------
---------	-------
---------	-------
---------	-------
---------	-------

Y2009

DTP_1	
id	---------
ds_id	----------
open	---------
high	---------
low	---------
close	----------
volume	decimal

DTP_10	
id	---------
ds_id	----------
open	---------
high	---------
low	---------
close	----------
volume	decimal

Y2010

Y2011

Y2019

Y2020

DTP_1	
id	----------
---------	----------
---------	----------
---------	----------
---------	----------
---------	----------

DIR

TL_1	
stock_id	int
symbol	char(10)

TL_2	
stock_id	int
symbol	char(10)

TL_10	
stock_id	int
symbol	char(10)

YEAR_	
yid	char(3)
year_col	year

MONTH_	
mid	char(3)
month_col	int

DAY_	
did	char(3)
day_col	int

TIMESTAMP_	
tid	char(5)
time_col	char(15)

The Schema||Table descriptions

DIR	DIR.YEAR_: yid is the index and PK given to years' 2008 up-to 2020. It maps to ds_id of all dtp_ tables in schemas YXXXX
	DIR.MONTH_: mid is the index and PK given to the 12 months. It maps to ds_id of all dtp_ tables in schemas YXXXX
	DIR.DAY_: did is the index and PK given to all possible dates, 1-31. It maps to ds_id of all dtp_ tables in schemas YXXXX
	DIR.TIMESTAMP: tid is the index and PK given to market timings 9:30AM to 4PM, every minute. It maps to ds_id of all dtp_ tables in schemas YXXXX
	DIR.**TL_1**: stock_id is the index and PK given to tickers of **bin 1**. It maps to ds_id of **dtp_1** tables in schemas YXXXX
	DIR.**TL_2**: stock_id is the index and PK given to tickers of **bin 2**. It maps to ds_id of **dtp_2** tables in schemas YXXXX

	DIR.**TL_10**: stock_id is the index and PK given to tickers of **bin 10**. It maps to ds_id of **dtp_10** tables in schemas YXXXX
YXXXX: [Y2008, Y2009, Y2010, Y2011, Y2012, Y2013, Y2014, Y2015, Y2016, Y2017, Y2018, Y2019, Y2020]	YXXXX.**DTP_1**: stores daily stock data of year XXXX for tickers in **bin 1** with an auto-incremented PK, **id**, on all rows
	YXXX.**DTP_2**: stores daily stock data of year XXXX for tickers in **bin 2** with an auto-incremented PK, **id**, on all rows
	YXXXX.**DTP_3**: stores daily stock data of year XXXX for tickers in **bin 3** with an auto-incremented PK, **id**, on all rows
	YXXXX.**DTP_4**: stores daily stock data of year XXXX for tickers in **bin 4** with an auto-incremented PK, **id**, on all rows
	YXXXX.**DTP_5**: stores daily stock data of year XXXX for tickers in **bin 5** with an auto-incremented PK, **id**, on all rows
	YXXXX.**DTP_6**: stores daily stock data of year XXXX for tickers in **bin 6** with an auto-incremented PK, **id**, on all rows
	----------
	YXXXX.**DTP_10**: stores daily stock data of year XXXX for tickers in **bin 10** with an auto-incremented PK, **id**, on all rows

For schemas **YXXXX the **ds_id** column is populated by mapping

ds_id: [date_id: yid+mid+did+tid] + [stock_id] in the directory tables.

**The ds_id (which is composed of numbers) is stored as CHAR because the date_id "numbers" have been represented as 01, 02, etc. *Why?*

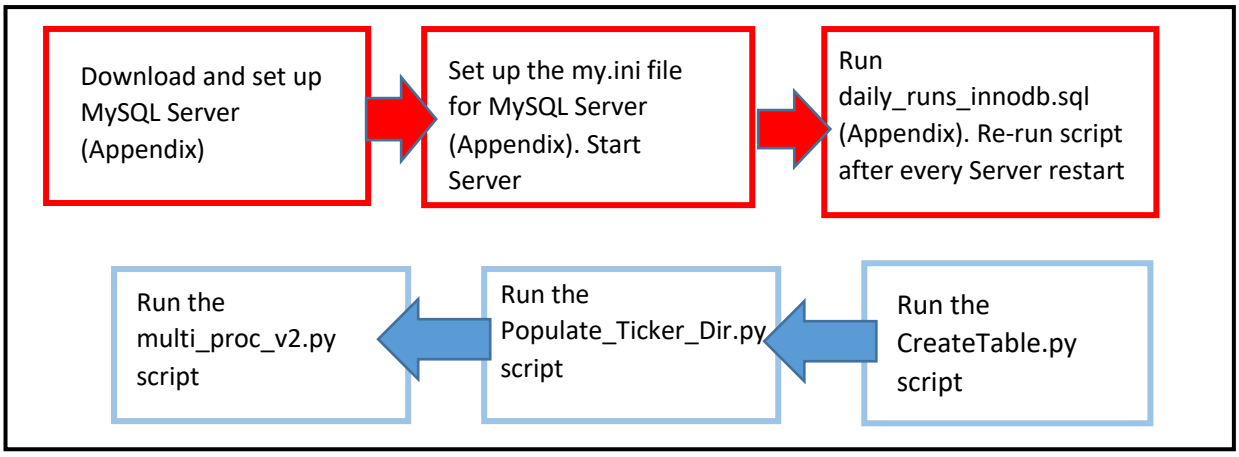
Because, not adding the 0's in front of ('0'1,'0'2, etc.) causes redundancy such as:

For stock number 5: ds_id: 11th January 2018 930am ds_id = 1111115

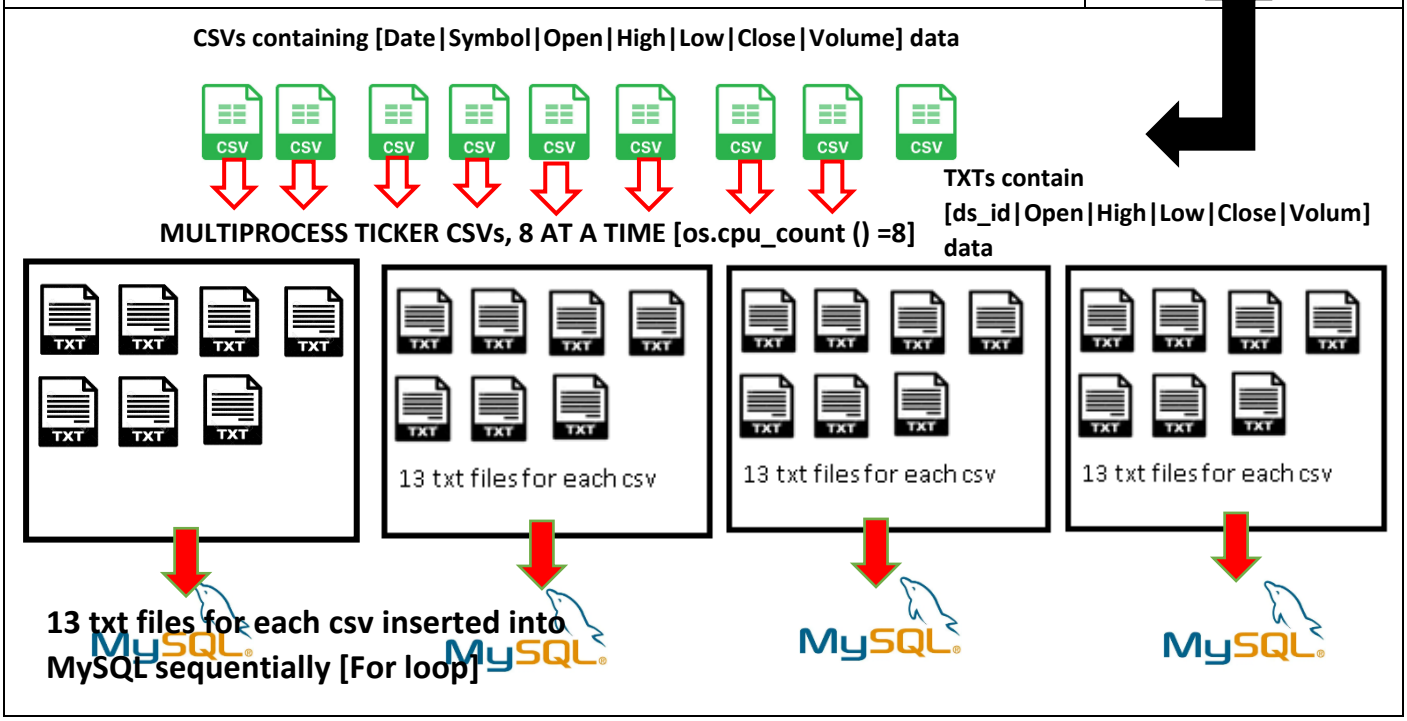
1st November 2018 930am ds_id = 1111115

Section 3

The Setup Flowchart



CreateTable.py Creates the entire MySQL structure for the schemas discussed, as per the **ERD** (Run instructions in Appendix)	CreateTable.py
Populate_Ticker_Dir.py Generates the 10 bins to distribute the ticker symbols into. Also, populates the ticker id directory tables **[tl_1, tl_2, tl_3, tl_4, tl_5, tl_6, tl_7, tl_8, tl_9, tl_10]** in the dir. schema. (Run instructions in Appendix)	Populate_Ticker_Dir.py
Multi_proc_v2.py [Run instructions in Appendix] Inserts the data-points for all tickers, from given input csv files into the respective tables **[dtp_1, dtp_2, dtp_3, dtp_4, dtp_5, dtp_6, dtp_7, dtp_8, dtp_9, dtp_10]** in the schemas [y2008, y2009, y2010, y2011, y2012, y2013, y2014, y2015, y2016, y2018, y2019, y2020]	multi_proc_v2.py

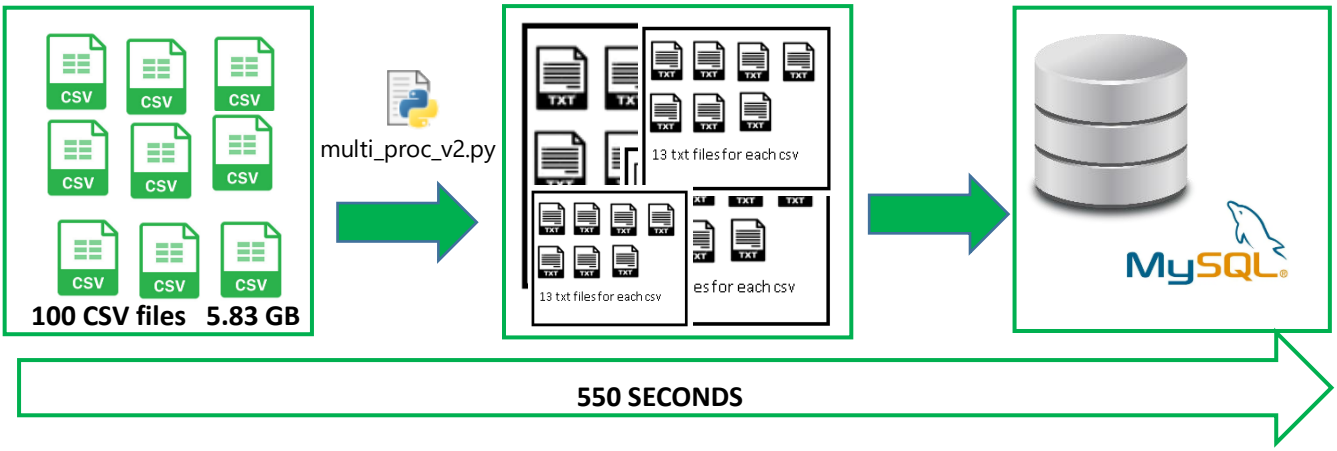


******For faster insert results into MySQL*

a. *Restart MySQL server*
b. *Run daily_runs_innodb.sql in MySQL Workbench*
c. *Only then run the multi_proc_v2.py script*

Section 4

The Methodology



Testing multi_proc_v2.py for 100 tickers (5.83 GB) gives a run-time of 550 seconds. The optimizations made are

- Inserting the data in txt format using MySQL command *LOAD DATA INFILE* (More in Appendix)
- Multiprocessing the csv files simultaneously in groups of 8 as per total available cpu
- Combining the dates and symbols into 1 column [ds_id], hence reducing data size along with removing requirement of a Foreign Key reference on ticker symbols. (Page 3)
- There is no Foreign Key reference on the ticker symbols in the dtp_ tables since the multi_proc_v2.py script is ensuring data integrity in two ways:
  - the ds_id in the dtp_ tables are mapped from the stock symbol id tables [tl_1, tl_2, etc.]
  - all the dtp_ table columns are declared NOT NULL in their table's respective DDL. A missing stock_id for any symbol will create a NAN value in the input txt file data.
  - So, if any stock_id and thus, ds_id is a NAN value, multi_proc_v2.py will throw an error

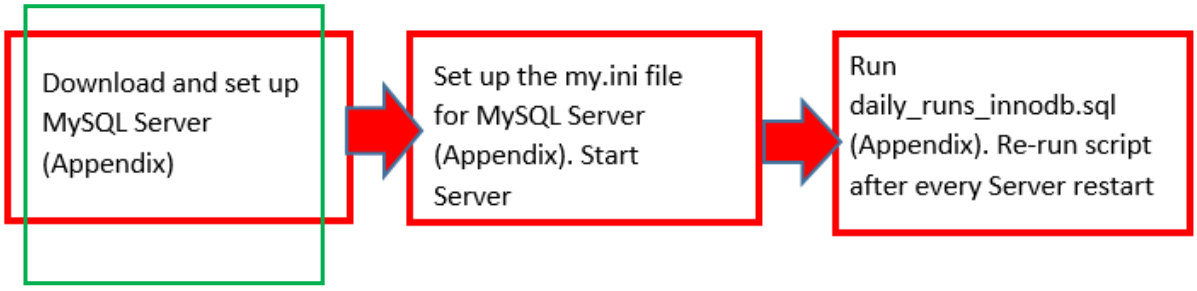
The issue(s) ahead

To add a reference directory in place in case a ticker's symbol changes in its input csv file. The changed symbol needs to be linked to the previous/original symbol such that they share the same stock_id

```
1606982916.6952403
1606982918.8148935
1606982922.8467212
1606982924.1136975
1606982932.1632614
1606982942.4507961
1606982948.9653907
Program took  554.1461536884308  seconds to run
Press any key to continue . . .
```

Capture of best run-time for multi_proc_v2.py

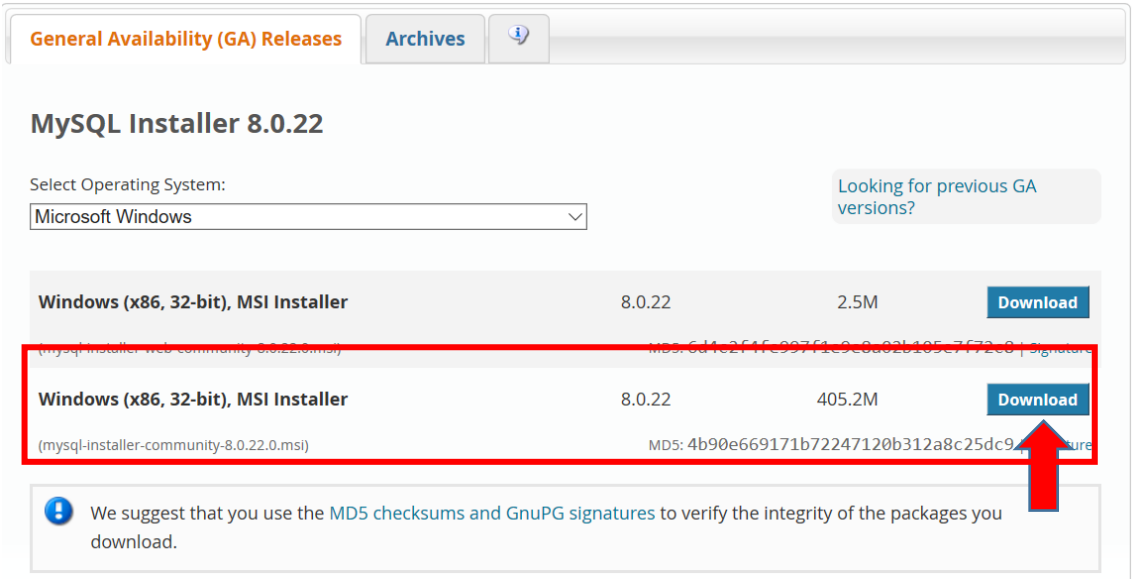
APPENDIX

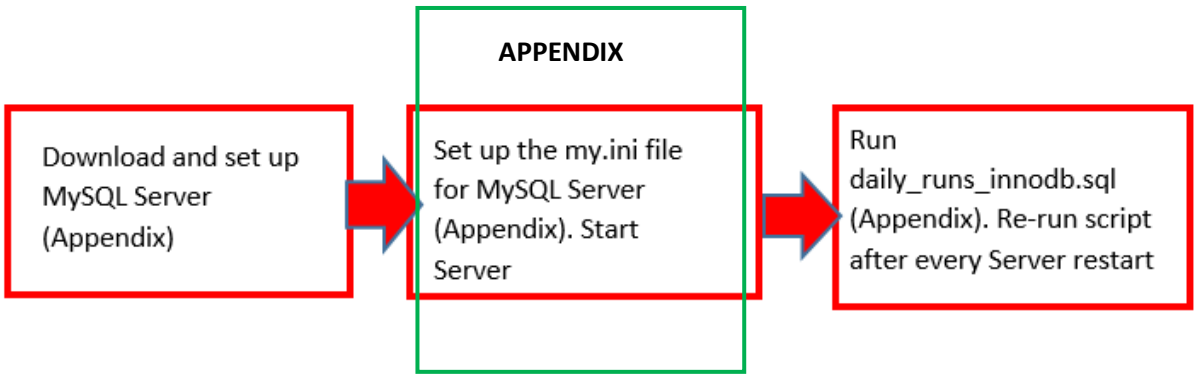


Download and set up MySQL Server

[Download link]

Follow the straightforward download instructions that pop-up when executing the installer on PC





Set up the my.ini file for MySQL Server

```
# InnoDB, unlike MyISAM, uses a buffer pool to cache both indexes and
# row data. The bigger you set this the less disk I/O is needed to
# access data in tables. On a dedicated database server you may set this
# parameter up to 80% of the machine physical memory size. Do not set it
# too large, though, because competition of the physical memory may
# cause paging in the operating system.  Note that on 32bit systems you
# might be limited to 2-3.5G of user level memory per process, so do not
# set it too high.
innodb_buffer_pool_size=11G

# The maximum size of one packet or any generated or intermediate string, or any parameter sent by the
# mysql_stmt_send_long_data() C API function.
max_allowed_packet=1G

# General and Slow logging.
log-output=NONE

general-log=0

general_log_file="DESKTOP-90HKQH5.log"

slow-query-log=0

slow_query_log_file="DESKTOP-90HKQH5-slow.log"

long_query_time=10

# Error Logging.
log-error="DESKTOP-90HKQH5.err"

# ***** Group Replication Related *****
# Specifies the base name to use for binary log files. With binary logging
# enabled, the server logs all statements that change data to the binary
# log, which is used for backup and replication.
# log-bin
skip-log-bin

# The TCP/IP Port the MySQL Server X Protocol will listen on.
loose_mysqlx_port=33060
innodb_doublewrite=0
```

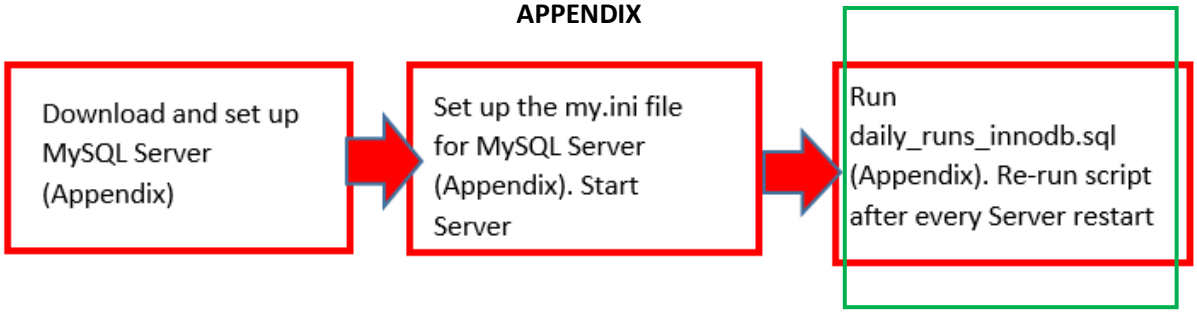
Set innodb_buffer_pool size between 60-80% of your PC RAM capacity.

Set the max_allowed_packet to maximum value i.e. 1G

Make all the highlighted changes, in-place, in the my.ini file – or as instructed and re-start the MySQL server after

Add this line at end of file

APPENDIX



Run daily_runs_innodb.sql

```
SET GLOBAL Innodb_change_buffering=none;
SET GLOBAL Innodb_log_buffer_size=1073741824;
SET GLOBAL Binlog_cache_size=262144;
#SET GLOBAL Innodb_log_file_size = 4294967296;
SET GLOBAL Innodb_io_capacity=2000;
SET GLOBAL Innodb_lru_scan_depth=8192;
SET GLOBAL Innodb_flush_log_at_trx_commit=0;
SET sql_log_bin=0;
SET GLOBAL Sync_binlog=0;
SET GLOBAL Innodb_checksum_algorithm=none;
SET GLOBAL Innodb_log_checksums=OFF;
SET GLOBAL Foreign_key_checks=0;
SET GLOBAL Unique_checks=0;
```

**Run the sql file on MySQL Workbench*

Run this file in MySQL Workbench after changing the my.ini file and starting the MySQL Server.

Run this file every time you re-start MySQL Server

Run multi_proc_v2.py ONLY after executing this file for faster INSERTS into MySQL

APPENDIX

Helper Files for MySQL Workbench

To run every time MySQL Server is re-started	daily_runs_innodb.sql
To run if need to drop the Dir & y2008-y2020 Schemas entirely	Drop_All_Schemas.sql
To run to truncate all dtp_1 to dtp_10 tables in all y2008-y2020 schemas	truncate_tables.sql
To run to truncate the ticker id tables tl_1 to tl_10 in the Dir schema	truncate_ticker_directory_tables.sql

Python script instructions for local run

CreateTable.py

```
conn = mysql.connector.connect(user='akanksha', password='
```

Change this line in script: user and password to your own MySQL Server login credentials and RUN script

Populate_Ticker_Dir.py

Change this line: the user and password to your own MySQL Server login credentials

```
connection = mysql.connector.connect(host='localhost',
                                     port = 3306,
                                     user='akanksha',

if __name__ == "__main__":
    print('Hello')
    # Point to the directory of inputs; stock market data CSV files
    os.chdir('C:\\Users\\akank\\Desktop\\Google Drive')
```

Change this line: os.chdir() to the folder which has all your stock input csv files and RUN script

Multi_proc_v2.py

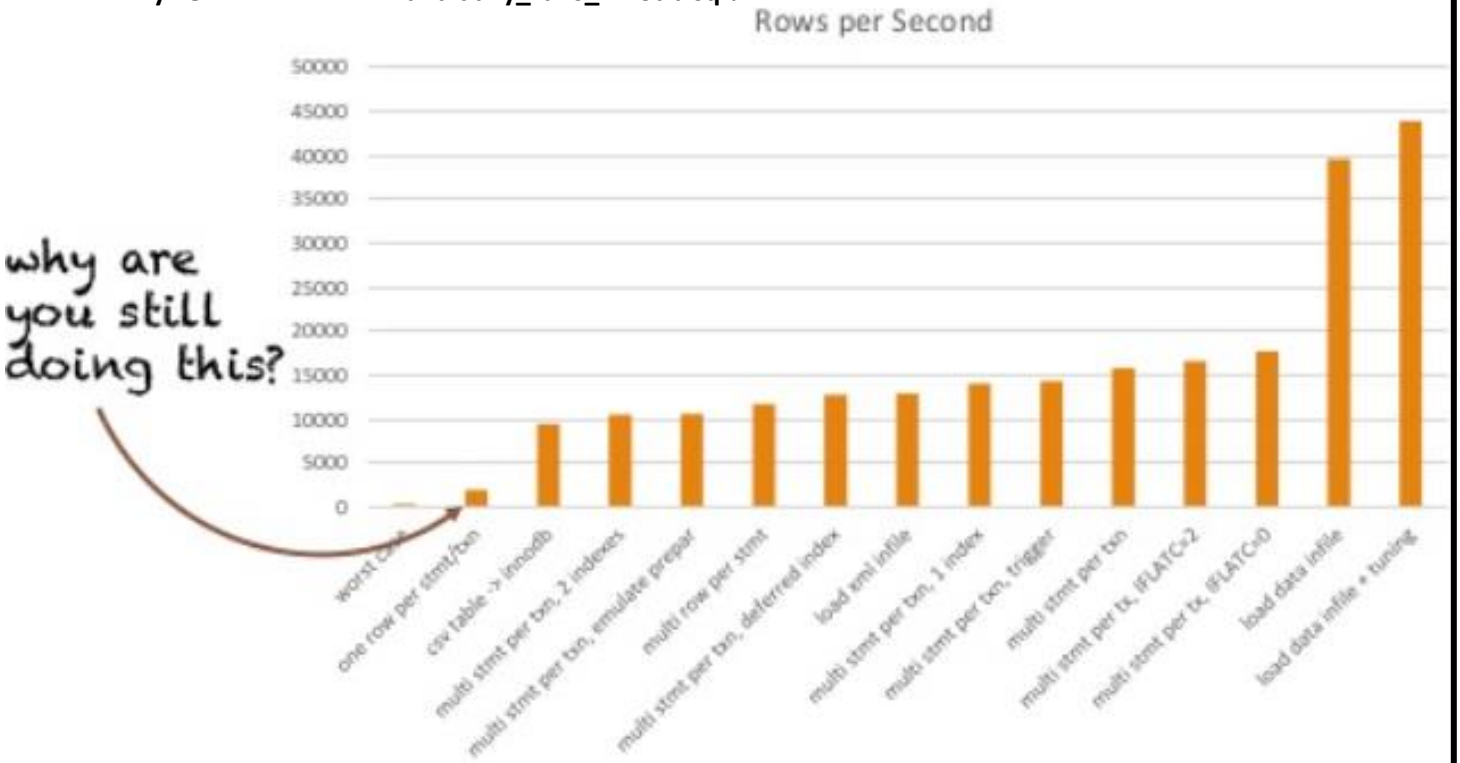
Change this line: the user and password to your own MySQL Server credentials

```
connection_ = mysql.connector.connect(host='l
                                      port = 3306,
                                      database='di
                                      user='akanks
                                      password='Ak
cursor = connection_.cursor()
engine = db.create_engine('mysql+mysqlconnect
connection = engine.connect()

if __name__ == '__main__':
    os.chdir('C:\\Users\Dell\Desktop\Google Drive')
    start_time = time.time()
```

Change this line: os.chdir() to the folder which has all your stock input csv files and RUN script

Why LOAD DATA INFILE and daily_runs_innodb.sql?



The LOAD DATA statement reads rows from a text file into a table at a very high speed. LOAD DATA is the complement of SELECT … INTO OUTFILE. Along with daily_runs_innodb.sql file's tuning statements, it helps achieve the best INSERT performance [category last column of graph]. The statements help improve InnoDB engine's performance, durability, compatibility and efficiency.

References

https://www.slideshare.net/billkarwin/load-data-fast

https://www.slideshare.net/zhaolinjnu/inno-db-internals-innodb-file-formats-and-source-code-structure

Procedure Summary

1. Download MySQL installer (Page 6)
2. Customize and setup MySQL CONFIG file in C:\ProgramData\MySQL\MySQL Server 8.0\my.ini (Page 7)
3. Run daily_runs_innodb.sql in MySQL Workbench (Page 8 and 9)
4. Set the MySQL user and password credentials in CreateTable.py, Populate_Ticker_Dir.py and multi_proc_v2.py (Page 9)
5. Set os.chdir() in Populate_Ticker_Dir.py and multi_proc_v2.py to the local folder containing csv input files (Page 9)
6. Run the scripts in sequence (Page 4)