**Pushing Data to MySQL**

1. Get the Data from Redis. Continuously watch if data has existed. When data has existed then follow below steps
2. This will also be configured in Scheduler. Once WebSocket is running and data is fetched then this also should at same time. We need to uses same schedular details.
3. Data should get insert as quickly as possible.
4. First Data should insert in FIFO method only.
5. Once 1 record in inserted then only that row should be delete from Redis
6. Name of table should **data\_harvesting\_20241118** format (current day format YYYYMMDD)
7. LastUpdateTime should be parsed from UNIX timestamp to IST (Please check logic in indexfuture.py
8. Trading symbol from mapping those exchangeInstrumentID and then insert (Please ask logic to Deepak or in same (indexfuture.py)
9. Once all records Redis is inserted in Table then Check the records count of table and data inserted. Both should be same. Raise Error on Console if both are not matching.
10. Check **“insert\_trade\_data**” Method from indexfuture.py
11. Add on more column “InsertedDateTime.
12. Map Quantity Column properly
13. Scheduler
14. Also add logic uk zone
15. async def insert\_trade\_data(data,combined\_token\_names):
16. try:
17. # Establish the database connection
18. #print(f'combined\_token\_names :{combined\_token\_names}')
19. connection = mysql.connector.connect(
20. \*\*DB\_CONNECTION\_PARAMS
21. )
22. table\_namenew = "data\_harvesting" + "\_" + datetime.now().strftime("%Y%m%d")
23. exchange\_id = data.get('ExchangeInstrumentID')
24. token\_name = None
25. time\_difference = 0
26. config = load\_config('config.json')
27. if config["timezone"].lower()=="uk":
28. time\_difference = 60
29. else:
30. time\_difference= 330
32. if isinstance(combined\_token\_names, dict):
33. token\_name = combined\_token\_names.get(exchange\_id, None)
34. elif isinstance(combined\_token\_names, list):
35. token\_name = next((item for item in combined\_token\_names if item.get('ExchangeInstrumentID') == exchange\_id), None)
37. if connection.is\_connected():
38. cursor = connection.cursor()
40. # SQL insert statement
41. insert\_query = f"""INSERT INTO {table\_namenew} (
42. MessageCode,
43. ExchangeSegment,
44. ExchangeInstrumentID,
45. BookType,
46. XMarketType,
47. LastTradedPrice,
48. LastTradedQuantity,
49. LastUpdateTime,
50. PercentChange,
51. Close,
52. Tradingsymbol,
53. OverallData
54. ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"""
56. # Prepare the data as a tuple
57. record = (
58. data['MessageCode'],
59. data['ExchangeSegment'],
60. data['ExchangeInstrumentID'],
61. data['BookType'],
62. data['XMarketType'],
63. data['LastTradedPrice'],
64. data['LastTradedQunatity'],
65. datetime.fromtimestamp(data['LastUpdateTime'] + 315532800) - timedelta(minutes=time\_difference) if data['LastUpdateTime'] is not None else None,
66. data['PercentChange'],
67. data['Close'],
68. get\_token\_name(exchange\_id, combined\_token\_names),
69. json.dumps(data)  # Ensure item is a dictionary or JSON-serializable object
70. )
72. # Execute the insert statement
73. cursor.execute(insert\_query, record)
75. # Commit the transaction
76. connection.commit()
78. except mysql.connector.Error as error:
79. print(f"Failed to insert record: {error}")
80. finally:
81. if connection.is\_connected():
82. cursor.close()
83. connection.close()
84. #print("MySQL connection is closed")