**Part 2**

Based on the raw data extracted from the previous section, come up with the ETL design to ingest and maintain this data in the Data Lake and the DWH. Design a sample schema based on this data to facilitate any analysis and answer the sample queries mentioned below

Expected output:

* List all the components involved in ETL strategy, refresh type, refresh frequency, tools/technologies, etc
* Define the strategy for streaming data and batch data coming in from this source for historical and near-real time analysis.
* Sample model/schema with Facts and Dimensions to help answer sample questions below.
* Make assumptions when necessary and list/describe the assumptions.

At a high level, the workflow of this project would look like the following:

1. **AWS Lambda** will perform API calls. Scheduling could be handled by Amazon EventBridge to pull data **every minute** to fetch data from the API.
2. If we’re agnostic of the crypto-pairs, data can be stored directly into **S3**. However, if we’d like to manage and organize data based on pairs (for example), we could leverage **AWS Kinesis Firehouse** to carefully allocate data into respective **S3** Buckets or locations.
3. **AWS Glue** could be used to perform ETL on the data to prepare it for loading into RedShift. I’m not a big fan of Glue because it tends to become very expensive; this job could be handled by another scheduled **Lambda** job, depending how often we’d like to upload to Redshift. Lambda would be a great option for batch operations. As part of this job we’d like to perform any computations and fields that would otherwise be an overhead burden downstream, such as breaking dates apart into day, week, year, hour, weekday/weekend components, etc.
4. **Redshift** is a great OLAP option; it’s built for analysis and is completely managed by AWS.

**Sample Questions:**

1. What is the 52 week high and low for all the items traded in the past 3 months?
2. What is the high/low price/volume in the past 2 hours?
3. What is the volume for a given timeframe?
4. Monthly, quarterly and yearly volume for items that have 10 million+ in volume over the past year (Make assumption that you have also downloaded data for multiple trade pairs).

**Answers**

These are example queries.

1.

-- find items trade in past 3 months, select only those for the report

WITH three\_month\_items AS(  
 SELECT  
 DISTINCT crypto\_id  
 FROM fact\_trade\_history  
 WHERE trade\_month >= *NOW*() - INTERVAL 3 MONTH

)  
SELECT  
 crypto\_id  
 , *MAX*(high\_price) AS 52\_week\_high  
 , *MIN*(low\_price) AS 52\_week\_low  
FROM fact\_trade\_history  
WHERE pairs in (SELECT crypto\_id from three\_month\_items)  
GROUP BY pairs;

2.

SELECT   
 crypto\_id  
 , *MAX*(high\_price) AS 2\_hour\_high  
 , *MIN*(low\_price) AS 2\_hour\_low  
FROM fact\_trade\_history  
WHERE trade\_hour >= *NOW*() - INTERVAL 2 HOUR;

3.

**Note:** I’m arbitrarily choosing start of day Feb 1 and Start of day Feb 2.

SELECT *SUM*(volume) AS total\_volume  
FROM fact\_trade\_history  
WHERE trade\_timestamp BETWEEN '2025-02-01 00:00:00' AND '2025-02-02 00:00:00';

4.

**Example Result**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trade Pair | Month | Quarter | Year | Monthly Volume | Quarterly Volume | Annual Volume |
| BTC-USD | 1 | 1 | 2024 | 3604500 | 10398035 | 124776420 |
| BTC-USD | 2 | 1 | 2024 | 3810234 | 10398035 | 124776420 |
| BTC-USD | 3 | 1 | 2024 | 2983301 | 10398035 | 124776420 |
| BTC-USD | 4 | 2 | 2024 | 4104120 | 11200988 | 124776420 |
| BTC-ETH | 1 | 1 | 2024 | 95323 | 2859690 | 7276448 |

**Strategy**

This is a complex query! First, we’ll find the cases within the last year in which volume >= 1million. Then we’ll filter to only include those cases. We shall use multiple partition functions to slam-dunk this all in one go rather than joining three tables. Interestingly, `spend` must be part of the group-by statement.

WITH annual\_trade\_volume AS (  
 SELECT  
 crypto\_id  
 , *SUM*(volume) AS total\_volume\_last\_year  
 FROM fact\_trade\_history  
 WHERE timestamp >= *NOW*() - INTERVAL '1 YEAR'  
 GROUP BY trade\_pair  
 HAVING *SUM*(volume) >= 10000000   
)  
  
/\*  
Note: if we had not previously separated MONTH, YEAR, etc. in our ETL, we would need to continuously execute `EXTRACT(YEAR FROM timestamp) AS year` (etc.) in our query, making it HIGHLY NON-SARGABLE. So we'll assume the columns have already been created.  
\*/  
SELECT  
 crypto\_id  
 , trade\_month  
 , *EXTRACT*(QUARTER FROM timestamp) AS trade\_quarter  
 , trade\_year  
 , *SUM*(volume) OVER (PARTITION BY trade\_pair, trade\_month) AS monthly\_volume  
 , *SUM*(volume) OVER (PARTITION BY trade\_pair, *EXTRACT*(QUARTER FROM timestamp)) AS monthly\_volume  
 , *SUM*(volume) OVER (PARTITION BY trade\_pair, trade\_year) AS yearly\_volume  
FROM fact\_trade\_history  
WHERE  
 crypto\_id IN (SELECT crypto\_id FROM annual\_trade\_volume)  
 AND timestamp >= *NOW*() - INTERVAL '1 YEAR'  
GROUP BY  
 crypto\_id

, volume  
 , trade\_year  
 , trade\_month  
 , trade\_quarter  
ORDER BY  
 crypto\_id,  
 trade\_year,  
 trade\_month;

**Expected Output:**

* Data model with Facts and Dimensions to help answer the above SQL questions
* List all the fields in dimension and fact tables.

A screenshot of a computer

Description automatically generated

**Solution:** Below is the code I used to develop the data model above. In our sample queries, we can easily rely on only using the fact\_trade\_history table until we need to left-join crypto\_name from the dim\_crypto table.

**Code:**

Table fact\_trade\_history as trade {

trade\_id int

customer\_id int

crypto\_id int

exchange\_id int

trade\_date date

trade\_month int

trade\_year int

trade\_time timestamp

trade\_type varchar

trade\_volume int

trade\_price int

total\_trade\_value int

transaction\_fee int

Indexes {

(trade\_id) [pk]

(customer\_id)

(crypto\_id)

(trade\_date)

}

}

Table dim\_customer as customer {

customer\_id int

customer\_name varchar

age int

gender varchar

email\_address varchar

monthly\_income int

occupation varchar

city varchar

state varchar

country varchar

home\_owner varchar

Indexes {

(customer\_id) [pk]

}

}

Table dim\_crypto as crypto {

crypto\_id int

crypto\_name varchar

date\_created date

product\_name varchar

sub\_category varchar

category varchar

price int

Indexes {

(crypto\_id) [pk]

}

}

Table dim\_exchange as exchange {

exchange\_id int

exchange\_name varchar

country varchar

fees decimal(5, 2)

Indexes {

(exchange\_id) [pk]

}

}

Ref: trade.customer\_id > customer.customer\_id

Ref: trade.crypto\_id > crypto.crypto\_id

Ref: trade.exchange\_id > exchange.exchange\_id