**Requirements – Simple Daily Statistics Web Tool:**

1. In brief, a simple python based webengine which displays few stats information from either input flatfiles (located on local or remote server) or form table in MySQL DB. That’s it.
2. Input file or DB table (same info):

**Fruits.txt:**

**Fruit, Price, Color**

Apple, 1.05, Red

Apple, 1.10, Green

Mango, 2.0, Yellow

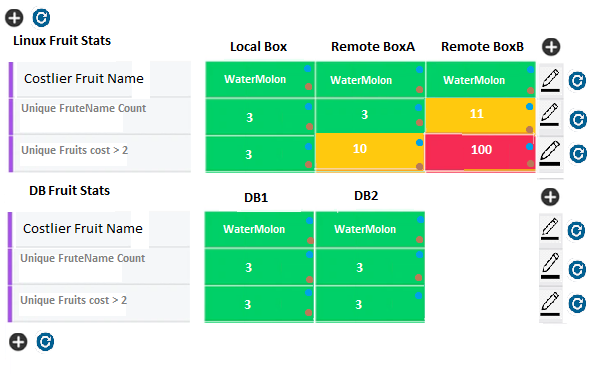
Mango, 2.0, Green

WaterMelon, 5.0, Red

Select \* from Fruits

|  |  |  |
| --- | --- | --- |
| **Fruit** | **Price** | **Color** |
| Apple | 1.05 | Red |
| Apple | 1.10 | Green |
| Mango | 2.0 | Yellow |
| Mango | 2.0 | Green |
| WaterMelon | 5.0 | Red |

1. **Required Web output:**

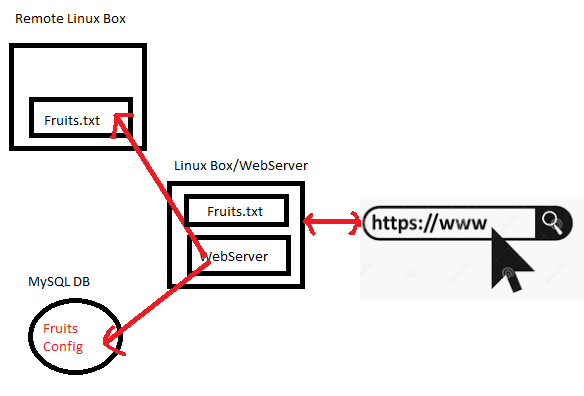
****

1. For stats example ( called Stats Points or rules – stored in DB):
2. How many unique fruits are available? Ans: 3 ( Apples,Mango,WaterMelon)
3. What is costliest Fruit name? Ans: WaterMelon
4. What is costliest Fruit Price ? Ans: 5.0
5. What is Price of Green Color Apple? 1.10
6. How many are different fruits available between price 1.5-3? 1 (Mango)

Important Points:

* 1. User Authentication/Role based
  2. On Demand – triggered by users:
     1. These ‘stats rules’ should be accepted from webpage and stores in MySQL DB (user will enter these configuration rules)
     2. User should able to view/modify these rules
     3. User should able to run these rules (all or selective rules)
     4. Web engine displays results from stored results on startup with last run timestamp details
     5. Results table will be created everyday like results\_YYYYMMDD
     6. Every time user runs ‘rule run’ command then results will be pushed results table so that other users can view same result on start-up (or new user can run command which will overwrite results for this rule)
     7. Results table should hold results for each and every rule
     8. Web engine should run results one by one when multiple users open trigger simultaneously ( so queuing system is required here)
     9. Its better if data or results is communicated with josn format so that curl command option be used
     10. Simple audit logs are required to store in audit table (logging purpose) like what is being modified and who triggered with timestamp
  3. Scheduled basis – triggered by web engine itself by looking up in table for scheduled jobs:
  4. Curl command option to run/view/modify/trigger rules instead of viewing
  5. Requirement should support max two mysql DB and 8 linux boxes
  6. Please use minimum plug-ins

**Diagram:**

****

**Input Files or DB:**

1. **Flat File (local or remote):**

Location file: either local Linux box where webserver resides

Or in remote server ( need to ssh command to access)

Content:

**Fruits.txt:**

**Fruit, Price, Color**

Apple, 1.05, Red

Apple, 1.10, Green

Mango, 2.0, Yellow

Mango, 2.0, Green

WaterMelon, 5.0, Red

1. Database (MySQL):

Select \* from Fruits

|  |  |  |
| --- | --- | --- |
| **Fruit** | **Price** | **Color** |
| Apple | 1.05 | Red |
| Apple | 1.10 | Green |
| Mango | 2.0 | Yellow |
| Mango | 2.0 | Green |
| WaterMelon | 5.0 | Red |

1. **Web should provide a functionality to accept input as fields:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Header Name ( or Category)** | | Unique Header or category name where group of same stats will be displayed | Linux Fruit Details  DB Fruit Details  Daily Max Price Stats  Etc… |
| Name | | Stats rules name | Unique Friuit Names,  Costliest Fruit Name  Costliest Fruit Price |
| Source | Database | Select DB details form drop down ( two mysql DBs ) | DB1  DB2 ( configurable) |
| SQL Query | Select Count(name) from fruits  Select Name from fruits  Where |
| Shell command | Shell Command |  |
| Linux Box (8 boxes) |  |
| Results Data Type | | * Interger * Float * String/Text * Python list or dict | **( Generic json format will avoid these all types)** |
| Color Range | | Add rules : Red >100  Green:< 10  Oragne: >10 & <100  Text:  Geen: result equal to “TEXT”  Red: result not equal to “TEXT” |  |
| Schedule Run Config | | Similar to cron entry config:  <https://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/> |  |
| Email | | Email id if incase scheduled run results need tobe sent |  |

**Tables required:**

Single command database:

* Fruits
* Config or Rules
* Logs or Audit

One for each day under Results\_YYYMMDD

* Results

**Simple User Cases or sequence:**

UserA logins very first time ( and has admin right to add config)

Empty webpage display as no rules are setup

UserA adds a config rule which consist “What is costliest fruit name?”

Webengine saves rule in MySQL config/rules table

A log entry is added in Audit

Notify save is Successful in web

Now UserA runs same rule manually by clicking run or refresh

Website displays results “WaterMelon” beside rule with timestamp

UserB logins very first time

One rule “What is costliest fruit name?” is displayed with results from results table ( results are as run from user A above)

Now UserB runs same rule manually by clicking run or refresh

Website displays results “WaterMelon” beside rule with latest timestamp

UserC logins very first time

One rule “What is costliest fruit name?” is displayed with results from results table ( results are as run from user B above)

UserC edits “What is costliest fruit name?” to run its scheduled job every weekday at 5pm by entering cronstring and email id

Webengire scans rules at 5pm and run rules and stores results and send email

UserD logins very first time after 6pm

One rule “What is costliest fruit name?” is displayed with results from results table ( results are as run from schedule job setup 5pm)

Now UserC runs same rule manually by clicking run or refresh

Website displays results “WaterMelon” beside rule with latest timestamp

**Sample web:**

**Scenario1**: Setup New ‘stats point’

Step1: User clicks “+” button too add new stats point in empty webpage



Step2: A new window/popup/help menu opens to accept below fields