Data Warehousing Project: SQL Implementation & BI Presentation
-Li Huang

Section 1: Background and Requirements

FIND is a company that give home shoppers the opportunity to find furniture and home decors they like via AI-powered search technology, by bringing the customer directly to merchants' product websites, based on an image the consumer uploads or chooses from the FIND app or website.
For merchants, this allows them to get in front of consumers who are specifically seeking out their brand's aesthetic, giving them more share of voice with your target audience and ultimately delivering more relevant leads.
Customers can instantly search 100s of furniture sites at once to easily find shoppable products based on images you love. Users upload a photo or choose from our library and FIND delivers precise matches through the use of proprietary computer vision (artificial intelligence) algorithms.

Company FIND want to leverage user's involvement information, including merchant users and customer users. In the direction to better understand customer behavior and experience, to increase users, and to understand the feedback/review of users.
They want to visualize and analyze the history data they have collected, and also be able to answer questions about the data by queries in the SQL system. For example, related KPIs could involve:
What is the daily/monthly customer usage of app?
How do our customers distribute across states?
How many new contracts with merchants we have per year?
How many merchants has continued cooperation with FIND for more than 3 years?
What is the average merchants/customers review by day?
How is our key partner merchant's review has changed by year?
What is the sales we have created for each merchant up to today?
What is the daily customer vs sales ratio has changed?
How many new customers have been developed in the past week?
How many merchants are in contract in FIND up to today?

Architecture
4 dimensional table and 1 aggregate table

- The Customer Dimension displays the info table of Customers that logged in on company's app 'FIND' daily, in the table CustomerInfo.
- The MerchantInfo contains the data of the merchants that signed contracts with the company, to allow the customers go to their shopping interface through App FIND.
- CustomerReview table and MerchantReview table contain feedback on app FIND from customers and users are collected, which are rated 5 levels satisfaction from 1 to 5 as Review, along with written content as Feedback.

- SalesFact table is the aggregate table that contains merchant ID, name, daily customer number on the app, total merchant sales daily and then created sales per customer daily info for each merchant.

Table name	Column name	Data Type	Length	Primary Key	Constraints	Example Data Value
CustomerInfo	CustomerID	int		PK	NOT NULL	"1000000001"
	CustomerName	varchar	255		NOT NULL	"Harry Smith"
	YearMonthDate	DATE			NULL DEFAULT ' 20010101'	"2013-3-1"
	LoginStamp	DATETIME		PK	NOT NULL	"2013-3-1 12:10:30"
	LogoutStamp	DATETIME			NOT NULL	"2013-3-1 12:30:20"
	City	varchar	255			"Chicago"
	IpAddress	varchar	255			"192.201.302.110"
	State	varchar	255			IL'
MerchantInfo	MerchantID	int	10	PK	NOT NULL	"2000000001"
	MerchantName	varchar	255			"Walmart"
	YearMonthDate	DATE		PK	NOT NULL	"2013-3-1"
	ContractStartYear	INT			NOT NULL	"2011"
	ContractEndYear	INT			NOT NULL	"2013"
	Sales	NUMERIC	(10,5)		NULL DEFAULT 0	"300.22"
CustomerReview	CustomerID	int		PK	NOT NULL	"1000000001"
	YearMonthDate	DATE				"2013-3-1"
	Review	NUMERIC	(10,5)			"4"
	Feedback	VARchar	255			"Huge help in finding the furniture I want. Please add more suppliers."
MerchantReview	MerchantID	int		PK	NOT NULL	"2000000001"
	YearMonthDate	DATE				"2013-3-1"
	Review	NUMERIC	(10,5)			"5"
	Feedback	VARchar	255			"Sales increased significantly after cooperation with FIND"
SalesFact	MerchantID	int	10	PK	NOT NULL	"2000000001"
	YearMonthDate	DATE		PK	NOT NULL	"2013-3-1"
	DailyTotalSales	NUMERIC	(10,5)			"53400.45"
	DailyCustomerNo	int				"200"

Source File example (displaying 3 records for each table):

MerchantInfo:

MerchantID	MerchantName	YearMonthDate	ContractStartYear	ContractEndYear	Sales
2000000001	Furniture Market	3/1/13	2011	2015	5,000
2000000002	Vans	3/1/13	2014	2016	100
2000000003	PopArt	3/1/13	2013	2015	300

CustomerInfo:

CustomerID	CustomerName	YearMonthDate	LoginStamp	LogoutStamp	City	IpAddress
1000000001	Harry Smith	3/1/13	3/1/13 12:10	3/1/13 12:30	Chicago	47.71.133.50
1000000002	Ian Wicker	3/1/13	3/1/13 14:00	3/1/13 15:00	New York	154.169.160.246
1000000003	Jennifer Hans	3/1/13	3/1/13 1:00	3/1/13 3:00	Seattle	27.73.52.99

MerchantReview:

MerchantID	YearMonthDate	Review	Feedback
2000000001	3/1/13	5	thank you!
2000000002	3/1/13	4	well done
2000000003	3/1/13	5	nice cooperation

CustomerReview:

CustomerID	YearMonthDate	Review	Feedback
1000000001	3/1/13	4	love it!
1000000004	3/1/13	4	Useful app, pls add more selection of furnitures.
1000000005	3/1/13	5	Huge help in finding the furniture I want. Please add more suppliers.

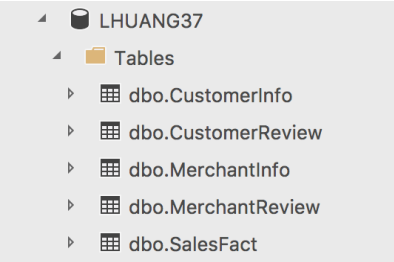
Section 2: SQL Implementation

(SQL code submitted in a separated file 'IS549 Final Project.sql'.)

There are 4 dimensional table and 1 aggregate table included in the project, and based on each one of them business related key facts and KPIs will be generated and analyzed in the following summary.

Part1. Create Dimensional tables and Aggregate Table and Populate the tables

- **Table structure:**



- **Table Content Review:**

1. Table CustomerReview

```
10    SELECT * FROM CustomerReview;
```

RESULTS

	CustomerID	YearMonthDate	Review	Feedback
1	1000000001	2013-03-01	4.00000	love it!
2	1000000004	2013-03-01	4.00000	Useful app, pls ...
3	1000000005	2013-03-01	5.00000	Huge help in fi...
4	1000000006	2013-03-02	2.00000	Not good matc...
5	1000000007	2013-03-02	3.00000	no comment
6	1000000007	2013-03-03	3.00000	Not helpful
7	1000000008	2013-03-03	4.00000	I like it.
8	1000000009	2013-03-02	5.00000	Best ever!
9	1000000009	2013-03-03	3.00000	Copying the ot...
10	1000000010	2013-03-03	5.00000	Great!

2. Table CustomerInfo

```
10    SELECT * FROM CustomerInfo;
```

RESULTS

	CustomerID	CustomerName	YMD	LoginStamp	LogoutStamp	City	IpAddress
1	1000000001	Harry Smith	2013-03-01	2013-03-01 12:...	2013-03-01 12:...	Chicago	47.71.133.50
2	1000000002	Ian Wicker	2013-03-01	2013-03-01 14:...	2013-03-01 15:...	New York	154.169.160.246
3	1000000003	Jennifer Hans	2013-03-01	2013-03-01 01:...	2013-03-01 03...	Seattle	27.73.52.99
4	1000000004	Emma Lee	2013-03-02	2013-03-02 17...	2013-03-02 19...	Boston	162.124.60.50
5	1000000005	Sara Wunder	2013-03-02	2013-03-02 12...	2013-03-02 12...	Washington D.C.	239.163.170.124
6	1000000006	Karissa Livoir	2013-03-02	2013-03-02 14...	2013-03-02 14...	Springfield	119.186.183.95
7	1000000007	Jeddy Livoir	2013-03-02	2013-03-02 15...	2013-03-02 16...	Philadelphia	102.215.89.162
8	1000000007	Jeddy Livoir	2013-03-03	2013-03-03 16...	2013-03-03 17...	Philadelphia	102.215.89.162
9	1000000008	Jacod Gerard	2013-03-03	2013-03-03 15...	2013-03-03 18...	Houston	194.54.132.145
10	1000000009	Caye Koch	2013-03-02	2013-03-02 15...	2013-03-02 15...	Los Angeles	187.241.88.185
11	1000000009	Caye Koch	2013-03-03	2013-03-03 00...	2013-03-03 01...	Los Angeles	187.241.88.185

3. Table MerchantInfo:

```
10    SELECT * FROM MerchantInfo;
```

RESULTS

	MerchantID	MerchantName	YearMonthDate	ContractStartYear	ContractEndYear	Sales
1	2000000001	Furniture Market	2013-03-01	2011	2015	5000.00000
2	2000000001	Furniture Market	2013-03-03	2013	2015	1000.00000
3	2000000002	Vans	2013-03-01	2014	2016	100.00000
4	2000000002	Vans	2013-03-03	2014	2016	300.00000
5	2000000003	PopArt	2013-03-01	2013	2015	300.00000
6	2000000003	PopArt	2013-03-02	2013	2015	500.00000
7	2000000004	V Sale	2013-03-02	2014	2015	2000.00000
8	2000000004	V Sale	2013-03-03	2014	2015	1000.00000
9	2000000005	Home Style	2013-03-02	2013	2016	1000.00000
10	2000000005	Home Style	2013-03-03	2013	2016	500.00000

4. Table MerchantReview

```
10  SELECT * FROM MerchantReview;
```

RESULTS

	MerchantID	YearMonthDate	Review	Feedback
1	2000000001	2011-03-01	4.00000	moderate impr...
2	2000000001	2012-03-01	4.00000	thanks!
3	2000000001	2013-03-01	5.00000	thank you!
4	2000000002	2013-03-01	4.00000	well done
5	2000000003	2013-03-01	4.00000	nice cooperation
6	2000000004	2013-03-01	4.00000	Good job
7	2000000005	2013-03-01	4.00000	decent increae ...

5. Aggregate Table: SalesFact

```
10  SELECT * FROM SalesFact;
```

RESULTS

	MerchantID	YearMonthDate	DailyTotalSales	DailyCustomerNo
1	2000000001	2013-03-01	5000.00000	3
2	2000000001	2013-03-03	1000.00000	4
3	2000000002	2013-03-01	100.00000	3
4	2000000002	2013-03-03	300.00000	4
5	2000000003	2013-03-01	300.00000	3
6	2000000003	2013-03-02	500.00000	5
7	2000000004	2013-03-02	2000.00000	5
8	2000000004	2013-03-03	1000.00000	4
9	2000000005	2013-03-02	1000.00000	5
10	2000000005	2013-03-03	500.00000	4

Part2. KPIs calculation and display

1. Based on Table CustomerInfo: Daily Unique Customer Number

```
10  SELECT YMD,COUNT(DISTINCT CustomerID) AS DailyCustomerNo FROM CustomerInfo
11  GROUP BY YMD;
```

RESULTS

	YMD	DailyCustomerNo
1	2013-03-01	3
2	2013-03-02	5
3	2013-03-03	4

2. Based on Table CustomerInfo: Total time Customer spend on the app per day

```
7   SELECT YMD, SUM(DATEDIFF(MINUTE,LoginStamp,LogoutStamp)) AS MinPerDay FROM CustomerInfo
8   GROUP BY YMD;
```

RESULTS

	YMD	MinPerDay
1	2013-03-01	200
2	2013-03-02	290
3	2013-03-03	410

3. Based on Table CustomerInfo: Average time customer spend on the app per day

```
6   SELECT a.YMD, a.MinPerDay/a.DailyCustomerNo AS AveragedCustomerTime
7   FROM(
8   SELECT YMD, COUNT(DISTINCT CustomerID) AS DailyCustomerNo,
9   SUM(DATEDIFF(MINUTE,LoginStamp,LogoutStamp)) AS MinPerDay
10  FROM CustomerInfo
11  GROUP BY YMD)a
```

RESULTS

	YMD	AveragedCustom...
1	2013-03-01	66
2	2013-03-02	58
3	2013-03-03	102

4. Based on Table CustomerInfo: Customer Location Count

```
10  SELECT State,COUNT(DISTINCT CustomerID) AS CustomerStateCount FROM CustomerInfo
11  GROUP BY State;
```

RESULTS

	State	CustomerStateCo...
1	CA	2
2	DC	1
3	IL	2
4	MA	1
5	NY	1
6	PA	1
7	TX	1
8	WA	1

5. Based on Table MerchantInfo: New Contracted Merchants by Year

```
10  SELECT ContractStartYear AS C_YEAR,COUNT(DISTINCT MerchantID) AS NewContracts FROM MerchantInfo
11  GROUP BY ContractStartYear;
```

RESULTS

	C_YEAR	NewContracts
1	2011	1
2	2013	3
3	2014	2

6. Based on Table MerchantInfo: Total merchant sales daily

```
7   SELECT MerchantID, YearMonthDate, SUM(Sales) AS DailyTotalSales FROM MerchantInfo
8   GROUP BY MerchantID, YearMonthDate;
```

RESULTS

	MerchantID	YearMonthDate	DailyTotalSales
1	2000000001	2013-03-01	5000.00000
2	2000000001	2013-03-03	1000.00000
3	2000000002	2013-03-01	100.00000
4	2000000002	2013-03-03	300.00000
5	2000000003	2013-03-01	300.00000
6	2000000003	2013-03-02	500.00000
7	2000000004	2013-03-02	2000.00000
8	2000000004	2013-03-03	1000.00000
9	2000000005	2013-03-02	1000.00000
10	2000000005	2013-03-03	500.00000

7. Based on Table MerchantInfo: Averaged Merchant sales daily

```
9   SELECT a.YearMonthDate, a.DailyTotalSales/a.DailyMerchantNo AS AveragedMerchantSales
10  FROM(
11  SELECT COUNT(DISTINCT MerchantID) AS DailyMerchantNo, YearMonthDate, SUM(Sales) AS DailyTotalSales
12  FROM MerchantInfo
13  GROUP BY YearMonthDate)a;
```

RESULTS

	YearMonthDate	AveragedMercha...
1	2013-03-01	1800.000000
2	2013-03-02	1166.666666
3	2013-03-03	700.000000

8. Based on Aggregate Table SalesFact, generate KPI DACS(DailyAveragedCustomerSales).

```
6   SELECT MerchantID, YearMonthDate, DailyTotalSales/DailyCustomerNo
7   AS DailyAveragedCustomerSales
8   FROM SalesFact;
```

RESULTS

	MerchantID	YearMonthDate	DailyAveragedCu...
1	2000000001	2013-03-01	1666.6666666...
2	2000000001	2013-03-03	250.00000000...
3	2000000002	2013-03-01	33.333333333...
4	2000000002	2013-03-03	75.000000000...
5	2000000003	2013-03-01	100.00000000...
6	2000000003	2013-03-02	100.00000000...
7	2000000004	2013-03-02	400.00000000...
8	2000000004	2013-03-03	250.00000000...
9	2000000005	2013-03-02	200.00000000...
10	2000000005	2013-03-03	125.00000000...

9. Based on table CustomerReview: Average CustomerReview

```
9   SELECT YearMonthDate, SUM(Review)/COUNT(CustomerID) AS AveragedCReview
10  from CustomerReview GROUP BY YearMonthDate;
```

RESULTS

	YearMonthDate	AveragedCReview
1	2013-03-01	4.333333
2	2013-03-02	3.333333
3	2013-03-03	3.750000

10. Based on table MerchantReview: Average merchant review

```
9   SELECT YearMonthDate, SUM(Review)/COUNT(MerchantID) AS AveragedMReview
10  from MerchantReview GROUP BY YearMonthDate;
```

RESULTS

	YearMonthDate	AveragedMReview
1	2011-03-01	4.000000
2	2012-03-01	4.000000
3	2013-03-01	4.200000

11. Based on table MerchantReview: Key client's review analysis

```
9   SELECT *
10  from MerchantReview WHERE MerchantID=2000000001;
```

RESULTS

	MerchantID	YearMonthDate	Review	Feedback
1	2000000001	2011-03-01	4.00000	moderate impr...
2	2000000001	2012-03-01	4.00000	thanks!
3	2000000001	2013-03-01	5.00000	thank you!

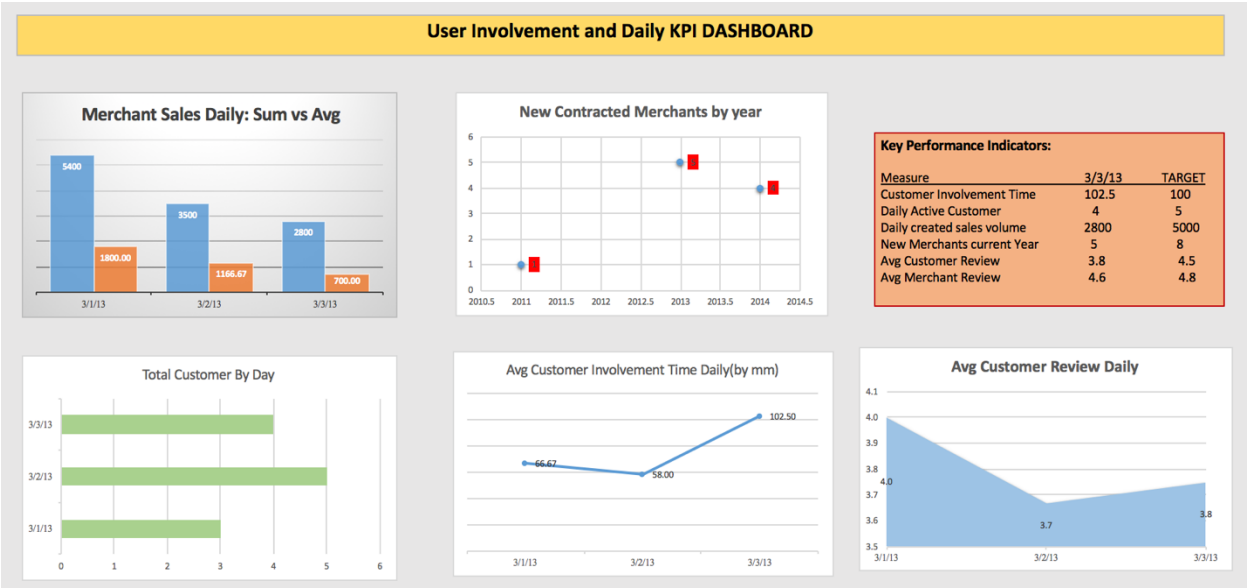
Section 3: BI representation

As shown above, I have mocked a dozen records of data for all my dimensional tables, and based on the data, created a dashboard and report for the strategy and marketing department. They can view the result on all devices, for example get a quick view from phone, but for better view and further investigation, laptop is recommended.

Dashboard

On the dashboard, I have present the KPI and key information that was required when we tried to develop the dimensional model.
The graph shown here are related with user involvement and the factors that indicating user loyalty. The top left is the values we have helped our contracted merchants to create, which is in two forms, total sales volume daily and average sales per merchant daily. The top right table is the new contracted merchant each year. The data is from table MerchantInfo.
The three graphs below are for the customer involvement and loyalty measurement. The left two is from the CustomerInfo table, it shown the total customer daily and average customer involvement time daily. The right table is from the table CustomerReview, it shows how the average customer review has changed from the past three days.

Here is a summary of the KPIs discussed above represented in dashboard:



Report
Since the limited number of mocked data, it would be hard to perform clustering or classification on the data, we will focus on certain points that are valuable to the business's strategy and marketing department.

- Satisfaction of Top 1% Customer

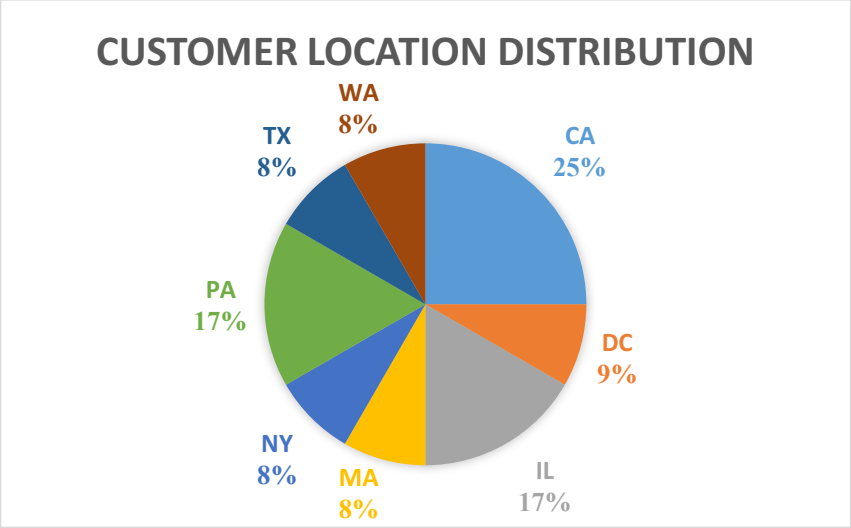
We are going to focus on the top 1% Customer's satisfaction by their review and feedback. Given our data amount, we are going to look at the two Customer That have spent the most time and money in the past three days. It is Customer 100000004 out of 10 customer listed here. This customer has rated us 4/5 and feedback is "Useful app, pls add more selection of furniture." From this review and feedback, we will know our top customer's need and possible ways we could improve. When the data is larger, the result will be more meaningful.

- Satisfaction of Top 20% Merchant

We would find our major profit source, the top 20% Merchant, it is merchant with ID 2000000001. We will look into the review from this merchant, which is 5/5, and find out which ways we can improve by talking to them directly. And we will look at the contract year from this merchant, which is 2011-2015(with current year 2013). We will talk to the merchant and try to help them increase sales and renew the contract.

- Development of new Customer

Let's look at Location distribution of Customer to help marketing department with their new customer development strategy.
We can see that our top customer is from state CA, PA, IL.



- Development for new Merchant

Let's look at the five data summary of sales volume fact for the past three days, and discover which types of merchants we have helped increased sales the most. It would be really helpful for marketing department and strategy department in developing new merchants, who are the source of profit of the company. We can see the maximum sales volume we have created is 5000, and the median is 750. This data could be used in marketing when we try to develop for new merchants.
And the Maximum sales is for the merchant "Furniture Market", the strategy department could study into similar merchants in the industry to develop potential new customers.

Merchant Three-day Sales Volume Summary

min	max	median	q1	q3
100	5,000	750	350	1000

Section 4: Requirements and Evaluation

As SQL (structured query language) is a programming language that enables programmers to work with data[1]. As we are using SQL to store the business related data and information, the business will be required to have SQL server build and storage for the data. At the same time, professional staff who know how to use SQL is also required. There may also be trainings for anyone who needs to query or use SQL inside business. And a basic suite of Microsoft offices including word, excel, PowerPoint is also required.

To evaluate the performance of SQL database, tables and queries, the business can use SQL server profiler[2] as start. And to look into it, there are several things we can look at[3]:

- Query Performance
  The most obvious place to look for poor query performance is in the query itself. Problems can result from queries that take too long to identify the required data or bring the data back. Look for these issues in queries: Selecting more data than needed or Inefficient joins between tables
- User and Query Conflicts
  Databases are designed to be multi-user, but the activities of multiple users can cause conflicts, for example, Page/row locking due to slow queries.
- Capacity
  Not all database performance issues are database issues. Some problems result from running the database on inadequate hardware, for example, not enough CPUs or CPU speed too slow.

From the previous SQL queries result, we did not come across any problem in running the database or queries. The query performed pretty fast, obviously enough capacity for the dozen examples in each table. There was no user and query conflicts occurred, everything goes smoothly. And the queries always returned what we asked for, no mistakes in performance. We can assume that the database, tables and queries up to now has met the criteria we have set as above.

To evaluate the performance of ETL software, there are some points to look into[4]:

- Architecture & infrastructure, for example:
  Symmetric Multiprocessing (SMP): Is Symmetric Multiprocessing supported? Standard in Windows NT and UNIX. The processors in SMP systems share their internal and external memory.
  Massively parallel processing (MPP): Every processor in a MPP system has its own internal and external memory and database, allowing high performance to be achieved. These databases should be synchronized.
  Cluster Aware: Is the ETL server ‘cluster aware’ and does it support load balancing, fail-over and other cluster facilities?
  Grid: Can an ETL process run on a ‘grid’ of computers or servers?

- Scalability, for example:
  Job distribution: Is it possible to run ETL processes on different machines or processors?
  Data pipelining: Can the different steps of an ETL process be run on different machines or processors?
  Partitioning: Is it possible to partition based on, for example, product codes, to determine on which machine or processor the data has to be processed?

- Core Functionality, for example:
  Splitting data streams/multiple targets: Is it possible to read a data source once and load the results into two or more tables?
  Union: Put rows of different tables with the same structure into one table or data set
  Conditional splitting: The same, but then in a condition, for example, if revenue is higher than 1000 put the results in table 1 otherwise in table 2

Cite References:

[1] https://learntocodewith.me/posts/sql-guide/
[2] http://www.sqlservercentral.com/articles/Video/66676/
[3] https://blog.appdynamics.com/engineering/top-6-database-performance-metrics-to-monitor-in-enterprise-applications/
[4] https://www.etltool.com/categories-and-criteria-examined/