**Data Warehousing Assignment**

**Scenario I**

In this scenario, we are interested in modelling student enrolment in Stanford courses. We would like to

answer questions such as:

• Which courses are most popular? Which instructors are most popular?

• Which courses are most popular among graduate students? Undergraduates?

• Are there courses for which the assigned classrooms are too large or too small?

We are planning to have a course enrolment fact table with the grain of one row per student per course enrolment. In other words, if a student enrols in 5 courses there will be 5 rows for that student in the fact table. We will use the following dimensions: Course, Department, Student, Term, Classroom, and Instructor.

There will be a single fact measurement column, Enrolment Count. Its value will always be equal to 1.

We are considering several options for dealing with the instructor dimension. Interesting attributes of instructors include FirstName, Last Name, Title (e.g., Assistant Professor), Department, and Tenured Flag. The difficulty is that a few courses (less than 5%) have multiple instructors. Thus, it appears we cannot include the instructor dimension in the fact table because it doesn’t match the intended grain. Here are the options under consideration:

**Option A:**

Modify the Instructor dimension by adding special rows representing instructor teams. For example, CS276ais taught by Manning and Raghavan, so there will be an instructor row representing “Manning/Raghavan” (as well as separate rows for Manning and Raghavan, assuming that they sometimes teach courses as sole instructors). In this way, the instructor dimension becomes true to the grain and we can include it in the fact table.

**Option B:**

Change the grain of the fact table to be one row per student enrolment per course per instructor. For example, there will be two fact rows for each student enrolled in CS 276a, one that points to Manning as an instructor and one that points to Raghavan. However, each of the two rows will have a value of 0.5 in the Enrolment Count field instead of a value of 1, in order to allow the fact to aggregate properly. (Enrolments are “allocated” equally among the multiple instructors.)

**Option C:**

Create two fact tables. The first has the grain of one row per student enrolment per course and doesn’t include the instructor dimension. The second has the grain of one row per student enrolment per course per instructor and includes the instructor dimension (as well as all the other dimensions). Unlike Option B, the value of Enrolment Count will be 1 for all rows in the second fact. Tell warehouse users to use the second fact table for queries involving attributes of the instructor dimension and the first fact table for all other queries.

**Please answer the following questions.**

**Question 1:**  What are the strengths and weaknesses of each option?

**Question 2:** Which option would you choose and why?

**Question 3:**  Would your answer to Question 2 be different if the majority of classes had multiple instructors? How about if only one or two classes had multiple instructors? (Explain your answer.)

**Question 4:** Can you think of another reasonable alternative design besides Options A, B, and C? If

so, what are the advantages and disadvantages of your alternative design?

**Solution:**

**Problem Statement:** We need to create a data model to track student enrolment in Stanford courses. We want to answer questions about popular courses, instructors, and classroom sizes. We plan to create a table with one row per student per course enrolment, but **We face a challenge: some courses have multiple instructors.**

**Example: Ram enrols in 5 courses. Each course has one instructor except for Course 5, which has two instructors.**

Here's how each option deals with the multiple instructor’s issue:

**Question 1: Strengths and weaknesses of each option:**

**Option A: Create special rows for instructor teams in the instructor column.**

Example:

| **Course** | **Instructor** |
| --- | --- |
| 1 | Instructor 1 |
| 2 | Instructor 2 |
| 3 | Instructor 3 |
| 4 | Instructor 4 |
| 5 | Instructor 5/6 |

* Strengths: Simple to understand, keeps one fact table, Instructor dimension true to grain.
* Weaknesses: Requires special rows for instructor teams, not efficient if many courses have multiple instructors.

**Option B: Change the table to have one row per student per course per instructor, and adjust the Enrolment Count.**

Example:

| **Course** | **Instructor** | **Enrolment Count** |
| --- | --- | --- |
| 1 | Instructor 1 | 1 |
| 2 | Instructor 2 | 1 |
| 3 | Instructor 3 | 1 |
| 4 | Instructor 4 | 1 |
| 5 | Instructor 5 | 0.5 |
| 5 | Instructor 6 | 0.5 |

* Strengths: Keeps one fact table, accommodates multiple instructors without special rows.
* Weaknesses: Changes fact table grain, Enrolment Count isn't a whole number (0.5), requires special handling.

**Option C: Create two separate fact tables, one without the instructor dimension, and another with the instructor dimension.**

**Fact Table 1 (without Instructor):**

| **Course** | **EnrollmentCount** |
| --- | --- |
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |

**Fact Table 2 (with Instructor):**

| **Course** | **Instructor** | **EnrollmentCount** |
| --- | --- | --- |
| 1 | Instructor 1 | 1 |
| 2 | Instructor 2 | 1 |
| 3 | Instructor 3 | 1 |
| 4 | Instructor 4 | 1 |
| 5 | Instructor 5 | 1 |
| 5 | Instructor 6 | 1 |

* Strengths: Separate fact tables allow for flexibility in querying, Enrolment Count is always 1.
* Weaknesses: Two fact tables may be confusing, requires users to choose correct table for queries.

**Question 2**: I would choose Option C because it provides the most flexibility and accuracy in the data model without complicating the fact table grain or Enrolment Count.

**Question 3:** If the majority of classes had multiple instructors, Option C would still be my choice due to its flexibility. If only one or two classes had multiple instructors, I might choose Option A for its simplicity, as there would be fewer special rows needed for instructor teams.

**Question 4:** Create a separate "Course Instructor" table to store relationships between courses and instructors.

Course Instructor Table:

| **CourseID** | **InstructorID** |
| --- | --- |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 5 | 6 |

Fact Table:

| **CourseID** | **Enrolment Count** |
| --- | --- |
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |

Advantages:

* Normalizes the relationship between courses and instructors, allowing multiple instructors per course.
* Keeps the fact table simple, with a consistent Enrolment Count of 1.
* No need for special rows or partial Enrolment Count values.

Disadvantages:

* Requires an additional table, which may add complexity to queries and joins.
* Users must join the Course Instructor table with the fact table to analyse instructor-related data.

**Scenario II:**

In this scenario, we are building a data warehouse for an online brokerage company. The company

makes money by charging commissions when customers buy and sell stocks. We are planning to have a Trades fact table with the grain of one row per stock trade.

**We will use the following dimensions:** Date, Customer, Account, Security (i.e. which stock was traded), and Trade Type.

**The company’s data analysts have told us that they have developed two customer scoring techniques that are used extensively in their analyses.**

1. Each customer is placed into one of nine Customer Activity Segments based on their frequency of transactions, average transaction size, and recency of transactions.
2. Each customer is assigned a Customer Profitability Score based on the profits earned as a result of that customer’s trades. The score can be either 1,2,3,4, or 5, with 5 being the most profitable.

**These two scores are frequently used as filters or grouping attributes in queries. For example:**

1. How many trades were placed in July by customers in each customer activity segment?
2. What was the total commission earned in each quarter of 2003 on trades of IBM stock by customers with a profitability score of 4 or 5?

There are a total of 100,000 customers, and scores are recalculated every three months. The activity level or profitability level of some customers changes over time, and users are very interested in understanding how and why this occurs.

**We are considering several options for dealing with the customer scores:**

**Option A:**

The scores are attributes of the Customer dimension. When scores change, the old score is over written with the new score (Type 1 Slowly Changing Dimension).

**Option B:**

The scores are attributes of the Customer dimension. When scores change, new Customer dimension rows are created using the updated scores (Type 2 Slowly Changing Dimension).

**Option C:**

The scores are stored in a separate Customer Scores dimension which contains 45 rows, one for each combi- nation of activity and profitability scores. The Trades fact table includes a foreign key to the Customer Scores dimension.

**Option D:**

The scores are stored in a Customer Scores outrigger table which contains 45 rows. The Customer dimensions includes a foreign key to the outrigger table (but the fact table does not). When scores change, the foreign key column in the Customer table is updated to point to the correct outrigger row.

**Please answer the following questions.**

**Question 5:** What are the strengths and weaknesses of each option?

**Question 6:** Which option would you choose and why?

**Question 7:** Would your answer to Question 6 be different if the number of customers and/or the time interval between score recalculations were much larger or much smaller? (Explain your answer.)

**Question 8:** Can you think of another reasonable alternative design besides Options A, B, C, and D?

If so, what are the advantages and disadvantages of your alternative design?

**Solution:**

**Problem Statement:** We are creating a data system for an online stock trading company. This company earns money when customers buy or sell stocks. We'll store this data in a table called "Trades" with information about each trade. We'll also use other tables to store information about dates, customers, accounts, stocks, and trade types.

The company's analysts use two scoring methods for customers:

1. Customers are divided into 9 groups based on how often they trade, the size of their trades, and when they last traded.
2. Customers get a score from 1 to 5, based on how much profit their trades generate for the company, with 5 being the most profitable.

These scores are important for analysing data, like finding how many trades happened in a month for each group or the total commission earned for specific stocks and customer profitability scores.

There are 100,000 customers, and their scores are updated every 3 months. It's important to track how these scores change over time.

**Question 5:**

**Option A: Overwrite old scores in the Customer table. When scores change, just update the values in the "Activity" and "Profit" columns. SCD 1**

| **Customer ID** | **Customer Name** | **Customer Activity Segment** | **Customer Profitability Score** | |
| --- | --- | --- | --- | --- |
| 1 | Ram | 3 | | 4 |
| 2 | Sita | 5 | 2 | |

Strengths:

* Simple to implement and understand.
* Saves storage space.

Weaknesses:

* Loses historical score data.
* Difficult to analyze score changes over time.

**Option B: Create new rows in the Customer table with updated scores. When scores change, create a new row with the updated "Activity" and "Profit" values. SCD 2**

**🡪 New row with updated scores**

| **Customer ID** | **Customer Name** | **Customer Activity Segment** | **Customer Profitability Score** | **Version** |
| --- | --- | --- | --- | --- |
| 1 | Ram | 3 | 4 | 1 |
| 1 | Ram | 4 | 5 | 2 |
| 2 | Sita | 5 | 2 | 1 |

Strengths:

* Preserves historical score data.
* Allows analysis of score changes over time.

Weaknesses:

* Increases storage space.
* More complex to manage and query.

**Option C: Store scores in a separate table and link it to the Trades table. When scores change, update the corresponding row in the Customer Scores table and link it to the Trades table.**

**Customer Table:**

| **Customer ID** | **Customer Name** |
| --- | --- |
| 1 | Ram |
| 2 | Sita |

**Customer Score Table:**

| **Score ID** | **Customer Activity Segment** | **Customer Profitability Score** |
| --- | --- | --- |
| 1 | 1 | 1 |
| 2 | 1 | 2 |

**Trades Fact Table:**

| **Trade ID** | **Date ID** | **Customer ID** | **Account ID** | **Security ID** | **Trade Type ID** | **Score ID** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 101 | 1 | 201 | 301 | 401 | 3 |
| 2 | 102 | 2 | 202 | 302 | 402 | 5 |

Strengths:

* Reduces redundancy in customer scores.
* Simplifies updating scores.

Weaknesses:

* Requires an additional table (Customer Scores).
* Slightly more complex to query.

**Option D: Store scores in a separate table and link it to the Customer table. When scores change, update the corresponding row in the Customer Scores table and link it to the Customer table.**

**Customer Dimension:**

| **Customer ID** | **Customer Name** | **Score ID** |
| --- | --- | --- |
| 1 | Ram | 3 |
| 2 | Sita | 5 |

**Customer Scores Outrigger Table:**

| **Score ID** | **Customer Activity Segment** | **Customer Profitability Score** |
| --- | --- | --- |
| 1 | 1 | 1 |
| 2 | 1 | 2 |

**Trades Fact Table:**

| **Trade ID** | **Date ID** | **Customer ID** | **Account ID** | **Security ID** | **Trade Type ID** |
| --- | --- | --- | --- | --- | --- |
| 1 | 101 | 1 | 201 | 301 | 401 |
| 2 | 102 | 2 | 202 | 302 | 402 |

Option D: Strengths:

* Reduces redundancy in customer scores.
* Simplifies updating scores.
* Keeps customer and score information separate.

Weaknesses:

* Requires an additional table (Customer Scores) and a link between Customer and Customer Scores tables.
* More complex to query.

**Question 6:**

I would choose Option B because it preserves historical score data and allows for analysis of score changes over time, which is important for the company's analysts.

**Question 7:**

If the number of customers and/or the time interval between score recalculations were much larger or smaller, my answer might be different. For a larger number of customers or more frequent recalculations, Option C or D could be more suitable to save storage space and simplify updates. For a smaller number of customers or less frequent recalculations, Option A might be sufficient since the impact of losing historical data would be less significant.

**Question 8:**

Alternative design: Option E - Hybrid approach Hybrid Slowly Changing Dimension (Type 6)

In this alternative option, the Customer dimension contains both current scores and historical scores for each customer. The current scores are stored in "Current Activity Segment" and "Current Profitability Score" columns, while the historical scores are stored in "Historical Activity Segment" and "Historical Profitability Score" columns. Additionally, there are "Effective Date" and "Expiry Date" columns to track the time period for which the historical scores were valid.

This approach allows us to maintain both current and historical data within a single row for each customer, making it simpler to query and analyse changes in customer scores. However, the table size will be larger compared to Option A but smaller than Option B.

Advantages:

* Preserves historical changes in customer scores.
* Simpler to implement and maintain compared to Option D.
* No additional join for queries involving scores.

Disadvantages:

* Larger dimension table size compared to Option A, but smaller than Option B.
* More complex queries when comparing current and historical scores.

**Customer Dimension:**

| **Customer ID** | **Customer Name** | **Current Activity Segment** | **Current Profitability Score** | **Historical Activity Segment** | **Historical Profitability Score** | **Effective Date** | **Expiry Date** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Alice | 4 | 5 | 3 | 4 | 2022-01-01 | 2022-06-30 |
| 1 | Alice | 4 | 5 | 4 | 5 | 2022-07-01 | 9999-12-31 |
| 2 | Bob | 5 | 2 | 5 | 2 | 2022-01-01 | 9999-12-31 |