Scenario: Designing a Star Schema from Raw Transaction Data

A Data Engineer at an e-learning platform receives a sample of raw data from course enrollment transaction logs, provided by the Backend team. The data is in a semi-structured format.
The task is to perform a manual ETL (Extract, Transform, Load) process to design an optimal Data Mart architecture in ClickHouse using a Star Schema.

Step 1: Extract

Study the sample raw data below. Each line represents a successful course enrollment.
Raw Data (Sample Enrollment Logs):

```
2025-07-11T09:05:12Z | event=enrollment_success | user_id=101;user_name=Budi S;user_city=Jakarta | course_id=C-01;course_name=Dasar-dasar Python;category=Pemrograman | price=550000;promo_code=HEMAT50
2025-07-11T09:07:45Z | event=enrollment_success | user_id=205;user_name=Citra A;user_city=Bandung | course_id=C-02;course_name=Manajemen Proyek Digital;category=Bisnis | price=750000;promo_code=NULL
2025-07-11T09:10:22Z | event=enrollment_success | user_id=101;user_name=Budi S;user_city=Jakarta | course_id=C-03;course_name=Desain Grafis untuk Pemula;category=Desain | price=600000;promo_code=DISKON10
```

Step 2: Transform (Data Modeling & Star Schema Design)

Begin the design of the star schema using paper or note-taking tools.

1. **Identify the Business Process (Fact)**:
   Specify the main event to be measured, which will serve as the fact table.
   - The central business event to measure is course enrollment. *Each log entry represents one successful course registration made by a user*.
   - Grain (Granularity): *One record per successful enrollment transaction*.
2. **Identify Dimensions**:
   Who, what, and when are involved in the process? These will become the dimension tables.
   Three key dimensions were identified based on the raw log structure:
   a. *User Dimension*: Captures user-related attributes.
   b. *Course Dimension*: Captures course-related attributes.
   c. *Time Dimension*: Derived from the timestamp of the enrollment.
   The raw data contains all the necessary fields to populate these dimension tables, along with the numeric values to be used as measures.
3. **Design the Fact Table:**
   - Define the name of the fact table.
     Name: ***fact_enrollment***.
   - List the columns, which should consist of foreign keys (linking to dimension tables) and measures (numeric values for analysis).
     The fact table stores each enrollment transaction. It includes foreign keys to the dimensions and numerical attributes for analysis.
     - Foreign Keys:
       a. ***time_id***: References *dim_time.time_id*.
       b. ***user_id***: References *dim_user.user_id*.
       c. ***course_id***: References *dim_course.course_id*.
     - Measures:
       a. ***price***: Original course price (integer, non-nullable).
       b. ***promo_code***: Discount code used during the transaction (nullable).

        c. ***final_price***: Nullable field to optionally capture the discounted amount (same type as price).

    The *final_price* column is intentionally nullable and not computed in this project scope. It is designed to support future enhancement if *price* adjustments based on *promo_code* are introduced.

4. **Design the Dimension Tables**:
    - For each identified dimension, create a table.
      Each dimension table captures descriptive attributes that relate to the fact records.
        a. ***dim_user***
            Primary Key: *user_id*
        b. ***dim_course***
            Primary Key: *course_id*
        c. ***dim_time***
            Primary Key: *time_id*
    - Define the table name and split the relevant attributes from the raw data into each dimension table. (Example: *user_name* and *user_city* should belong to the user dimension, not the fact table).
        a. ***dim_user***
           Columns:
            i. ***user_id***: Unique identifier for the user (*UInt64*).
            ii. ***user_name***: User's full name (*LowCardinality*(*String*)).
            iii. ***user_city***: City of the user (*LowCardinality*(*String*)).
        b. ***dim_course***
           Columns:
            i. ***course_id***: Unique identifier for the course (*LowCardinality*(*String*)).
            ii. ***course_name***: Course title (*String*).
            iii. ***category***: Course category (*LowCardinality*(*String*)).
        c. ***dim_time***
           Columns:
            i. ***time_id***: Original DateTime value from the event (*DateTime*).
            ii. ***date***: Extracted date (*Date*).
            iii. ***year***: Extracted year (*UInt16*).
            iv. ***month***: Extracted month (*UInt8*).
            v. ***day***: Extracted day (*UInt8*).
            vi. ***hour***: Extracted hour (*UInt8*).

5. **Define Optimization Keys**:
   For each table created in ClickHouse, consider the best strategy for PARTITION BY and ORDER BY.
    a. ***fact_enrollment***
        - Partitioning

```
PARTITION BY toYYYYMM(time_id)
```

          Divides the table by month based on the event timestamp, enabling partition pruning in queries filtered by date ranges.
        - Ordering

```
ORDER BY (course_id, time_id, user_id)
```

This ordering is designed for analytic workloads that involve time-based aggregations or course-specific breakdowns.

b. ***dim_user***

- Ordering

```
ORDER BY user_id
```

Provides efficient primary key lookups when joining with the fact table.

c. ***dim_course***

- Ordering

```
ORDER BY course_id
```

Aligns with the foreign key in the fact table to improve join performance and course-based filters.

- Column Optimization

```
LowCardinality(String)
```

Used on *course_id* and *category* columns to optimize storage and query performance by reducing dictionary lookup overhead for fields with few unique values.

d. ***dim_time***

- Ordering

```
ORDER BY time_id
```

Sorts by timestamp for efficient time-based filtering. *PARTITION BY* is not applied because the time table is relatively small and designed primarily for joining, not aggregation. Thus, *ORDER BY* alone is sufficient.

Step 3: Load

Prepare complete *CREATE TABLE* statements for all tables (fact and dimensions) in ClickHouse, based on the star schema design from Step 2.

- Schema Definition (DDL):

  Presents the DDL statements for implementing the star schema, with dimension tables created first to preserve logical integrity and enable fact table references.

a. ***dim_user***

```
CREATE TABLE IF NOT EXISTS dim_user (
    user_id UInt64,
    user_name String,
    user_city LowCardinality(String)
)
ENGINE = MergeTree()
ORDER BY user_id;
```

b. ***dim_course***

```
CREATE TABLE IF NOT EXISTS dim_course (
    course_id LowCardinality(String),
    course_name String,
    category LowCardinality(String)
)
ENGINE = MergeTree()
ORDER BY course_id;
```

c. ***dim_time***

```
CREATE TABLE IF NOT EXISTS dim_time (
    time_id DateTime,
    date Date,
    year UInt16,
    month UInt8,
    day UInt8,
    hour UInt8
)
ENGINE = MergeTree()
ORDER BY time_id;
```

d. ***fact_enrollment***

```
CREATE TABLE IF NOT EXISTS fact_enrollment (
    time_id DateTime,
    user_id UInt64,
    course_id LowCardinality(String),
    price UInt32,
    promo_code LowCardinality(Nullable(String)),
    final_price Nullable(UInt32)
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(time_id)
ORDER BY (course_id, time_id, user_id);
```

- Result Tables
  Displays the actual table schemas as created in ClickHouse, showing column order, data types, and applied optimizations as a result of the DDL execution.

a. ***dim_user***

```
user_id|user_name|user_city
-------+---------+----------
    101|Budi S   |Jakarta
    112|Lia F    |Jakarta
    205|Citra A  |Bandung
    310|Eka P    |Surabaya
    415|Farah D  |Medan
    520|Gilang R |Yogyakarta
    633|Hana L   |Semarang
    741|Indra K  |Denpasar
    852|Jasmine M|Makassar
    963|Kevin O  |Palembang
```

b. ***dim_course***

```
course_id|course_name               |category
---------+--------------------------+-----------
C-01     |Dasar-dasar Python        |Pemrograman
C-02     |Manajemen Proyek Digital  |Bisnis
C-03     |Desain Grafis untuk Pemula|Desain
C-04     |Pemasaran Media Sosial    |Pemasaran
C-05     |Analisis Data dengan SQL  |Pemrograman
```

c. ***dim_time***

```
time_id            |date      |year|month|day|hour
-------------------+----------+----+-----+---+----
2025-07-11 16:05:12|2025-07-11|2025|    7| 11|   9
2025-07-11 16:07:45|2025-07-11|2025|    7| 11|   9
2025-07-11 16:10:22|2025-07-11|2025|    7| 11|   9
2025-07-11 16:12:00|2025-07-11|2025|    7| 11|   9
2025-07-11 16:15:30|2025-07-11|2025|    7| 11|   9
2025-07-11 16:18:05|2025-07-11|2025|    7| 11|   9
2025-07-11 16:21:11|2025-07-11|2025|    7| 11|   9
2025-07-11 16:22:40|2025-07-11|2025|    7| 11|   9
2025-07-11 16:25:01|2025-07-11|2025|    7| 11|   9
2025-07-11 16:28:19|2025-07-11|2025|    7| 11|   9
2025-07-12 18:30:00|2025-07-12|2025|    7| 12|  11
2025-07-12 18:32:15|2025-07-12|2025|    7| 12|  11
2025-07-12 18:35:45|2025-07-12|2025|    7| 12|  11
2025-07-12 18:38:03|2025-07-12|2025|    7| 12|  11
2025-07-12 18:40:55|2025-07-12|2025|    7| 12|  11
2025-07-13 21:01:20|2025-07-13|2025|    7| 13|  14
2025-07-13 21:03:33|2025-07-13|2025|    7| 13|  14
2025-07-13 21:05:00|2025-07-13|2025|    7| 13|  14
2025-07-13 21:08:18|2025-07-13|2025|    7| 13|  14
2025-07-13 21:11:49|2025-07-13|2025|    7| 13|  14
```

d. ***fact_enrollment***

```
time_id            |user_id|course_id|price |promo_code|final_price
-------------------+-------+---------+------+----------+-----------
2025-07-11 16:05:12|    101|C-01     |550000|HEMAT50   |
2025-07-11 16:12:00|    310|C-01     |550000|          |550000
2025-07-12 18:30:00|    741|C-01     |550000|HEMAT50   |
2025-07-12 18:40:55|    205|C-01     |550000|          |550000
2025-07-11 16:07:45|    205|C-02     |750000|          |750000
2025-07-11 16:18:05|    415|C-02     |750000|          |750000
2025-07-12 18:32:15|    852|C-02     |750000|          |750000
2025-07-13 21:08:18|    633|C-02     |750000|          |750000
2025-07-11 16:10:22|    101|C-03     |600000|DISKON100 |
2025-07-11 16:25:01|    633|C-03     |600000|          |600000
2025-07-13 21:01:20|    112|C-03     |600000|DISKON100 |
2025-07-13 21:11:49|    741|C-03     |600000|          |600000
2025-07-11 16:15:30|    205|C-04     |700000|MARKET25  |
2025-07-11 16:28:19|    310|C-04     |700000|MARKET25  |
2025-07-12 18:35:45|    101|C-04     |700000|          |700000
2025-07-13 21:05:00|    520|C-04     |700000|MARKET25  |
2025-07-11 16:21:11|    520|C-05     |650000|SQLMASTER |
2025-07-11 16:22:40|    101|C-05     |650000|          |650000
2025-07-12 18:38:03|    963|C-05     |650000|SQLMASTER |
2025-07-13 21:03:33|    415|C-05     |650000|          |650000
```