

Project: Build a Data Warehouse - Phase 1

Monday, October 24, 2016 1:31 PM

Purpose: In Phase 1, converting an ERD (Dimensional Diagram) into an actual Data Warehouse database and fill it with data is a very important skill. Even more important is the ability to query a database for relevant information for the purposes of Strategic/Decisions Support reporting and changing values when necessary. This exercise will include tasks that develop such skills.

Warning:

Don't be afraid of making mistakes! It's OK to do so. In life misunderstandings are common, in fact it is the **ONLY** way we learn. It is **NOT** a waste of time to redo a drawing or part of a project if there is a misunderstanding. Let's be clear about this fact. If an instruction is ambiguous, just make a judgement call. I trust since you are in this upper-level course that you have developed or nearly developed decision-making skills. If not. There is no time like the present to learn :)

Second Warning:

READ THIS ENTIRE DOCUMENT. PRINT AND MAKE NOTES ON THIS DOCUMENT if necessary BEFORE starting ANYTHING. Have it in front of you throughout the *entire phase*. I will ask you for the hard copy of your notes on this document when you come see me about any questions. I will also expect a screen capture or picture of this document when you ask a question about the instructions. If you do not bring it or produce it, then, our conversation will be very short and only one instruction given-- Go get it and then come back OR Go produce it and then come back.

This assignment is divided into 3 sections:

1. **Data Definition**
2. **Data Transformation**
3. **Informational and Update Queries**

Required Items:

1. [ERD](#) Dimensional Model you constructed as result of the "Construct a Dimensional Model" assignment. (this item must be *completed* and **have a passing grade** BEFORE you begin!!)
2. **MS Access database called "ProductionDB-USELOG.mdb" attached to this assignment!** - Hereafter knows as **Production** database. It is the same, but I added some data to it to help you with your ETL data transformation queries. If you do NOT see a MAJORS table in the database, then download the correct copy.
3. *Read and make notes* on the [LABINFO Data Warehouse - Best Practices Model](#). This document will walk you through the organizational process and give you some additional items/examples that you may use as part of your submission.

Core assignment goal:

Use the specified ERD in the **Required Items** section to write SQL statements **ONLY** that will produce a database called "**LABINFO**" derived from transformed data in the **Production** database .

Work Product:

You will provide *hand-written* deliverables in the form of a MS Word document for **each section (3 sections = 3 files)** using the following instructions:

File 1:

Name: [Data Definition](#) (CREATE DATABASE, CREATE TABLE...)

1. *Write BY HAND* the SQL statements that construct (CREATE) a Data Warehouse **database** named "LABINFO" and **all** of the **tables** (entities and attributes) using the MySQL database language

syntax. The MySQL syntax was used in the embedded questions of the interactive slide sets. Review the slide sets if you need to refer back to that syntax.

2. Use the data types found in the required MS Access database called **Production** database. You should use *common sense* judgement when defining the data type of any new tables (dimensions). If the value uses any kind of text or character values, use the VARCHAR() data type. When using the VARCHAR data type make sure you define the maximum length potential for each value. For example: If you think the value will be on average 10 characters, may reach a maximum of 12, then use VARCHAR(12). For a list of data types you may use, consult the list of data types provided in the slide set.
3. Remember, **Primary Key** fields *require* the **NOT NULL** property.
4. Do not worry about the Foreign Key or UNIQUE properties for this exercise

File 2:

Name: **Data Transformation** (INSERT INTO and INSERT INTO...SELECT) ETL

1. Write **BY HAND** the **SQL statements only** that will **INSERT** derived data from the **Production** database into the new tables you created in the data definition section above for **LABINFO**. Refer to the answers you gave as part of the slide sets questions. Use INSERT INTO and/or INSERT INTO...SELECT statements to copy from the Production database to the target dimension table or fact table. Here is an **example**:

```
INSERT INTO DIM_MAJOR (MAJOR_CODE, MAJOR_NAME)
SELECT DISTINCT MJ.MAJOR_CODE, MJ.MAJOR_NAME
FROM production.STUDENT AS ST, production.MAJOR AS MJ
WHERE
ST.MAJOR_CODE = MJ.MAJOR_CODE;
```

2. For the USEFACT and each DIMENSION table, you may need to use both the INSERT INTO and INSERT INTO...SELECT statement with aggregate functions like COUNT, SUM to generate the proper totals for each dimension. You will also need to use the SELECT DISTINCT clause as well for creating the dimensional data. I will provide a separate *Best Practices Model (attached to the assignment)* for those struggling with this part of the process. Here is an example to fill the use-fact table

```
INSERT INTO USEFACT ( SEMESTER_ID, MAJOR_CODE, CLASS_ID,
TIME_ID, GROUP_PROJECT_ID, TOTAL_VISITS, TOTAL_TIME )
SELECT DISTINCT SEMESTER_ID, MAJOR_CODE, CLASS_ID, TIME_ID,
GROUP_PROJECT_ID, COUNT(STUDENT_ID) AS TOTAL_VISITS,
SUM(DURATION) AS TOTAL_MINUTES
FROM USELOG_ODS
GROUP BY SEMESTER_ID, MAJOR_CODE, CLASS_ID, TIME_ID,
GROUP_PROJECT_ID;
```

3. Writing these statements will **NOT** be trivial and will *require research from the textbook, slide-sets and World Wide Web*. You **WILL NEED to set aside at least 10 hours** for this section alone. For those in study groups, this is your chance to run ideas off of each other. But remember, I will know if you are copying from one another. And will impose proper academic sanctions to ALL PARTIES involved should I discover such activity .
4. Each row **should have a value** in *every attribute*-- despite absence the NOT NULL specification (FOR INSERT Statements).

File 3

Name: **Informational and Update Queries** (SELECT..., UPDATE..., DELETE...)

1. Write **BY HAND** two (2) SQL statements and a description of the desired result that refer to the data defined earlier in the LABINFO database for **each** of the following tasks:
 - a. **Selection queries**
 - i. Select query that returns **ALL columns** (attributes) in any table

- ii. Select query that returns **at least two (but not ALL) columns** in any table
- iii. Select query that returns **at least two (but not ALL) columns** and defines the selection criteria based on a WHERE clause
- iv. Select query that returns a different column than that which is specified in the WHERE clause
- v. Select query that returns at least one type of Join
- vi. Example:

```
SELECT DISTINCT CLS.CLASS_DESCRIPTION,
SUM(TOTAL_VISITS) AS TOT_VISITS, SUM(TOTAL_TIME) AS
TOT_MINUTES, SUM(TOTAL_TIME)/60 AS TOT_HOURS,
(SUM(TOTAL_TIME)/60)/SUM(TOTAL_VISITS) AS AVG_TIME
FROM USEFACT AS UF, DIM_MAJOR AS MJ, DIM_CLASS AS CLS,
DIM_GROUP_PROJECTS AS GP
WHERE UF.GROUP_PROJECT_ID = GP.GROUP_PROJECT_ID AND
UF.MAJOR_CODE = MJ.MAJOR_CODE AND
UF.CLASS_ID = CLS.CLASS_ID
GROUP BY CLS.CLASS_DESCRIPTION;"
```

b. DELETE|INSERT queries (to update LABINFO with potentially changed data from the Production database)

- i. DELETE|INSERT query that updates a dimension based on changes in the production database)
- ii. DELETE|INSERT that changes two or more values in any row(s) of a table
- iii. DELETE|INSERT that changes two or more values defined by a selection criteria using a WHERE clause
- iv. Example:

```
DELETE FROM CLASS;
INSERT INTO DIM_CLASS(CLASS_ID, CLASS_DESCRIPTION)
SELECT DISTINCT ST.CLASS_ID, CL.CLASS_DESCRIPTION
FROM PRODUCTION.STUDENT AS ST, PRODUCTION.CLASS AS CL
WHERE
ST.CLASS_ID = CL.CLASS_ID;
```

c. DO NOT reuse queries or WHERE clauses defined in earlier tasks. YOU WILL NOT get credit if every statement is not unique

2. Create 3 SQL Views in hand-written form. Use ANY combination of TABLES/COLUMNS that describes a specific task to *create a SQL View*. At least two of the views must of have joins similar to example task 2.

a. Example (DO NOT USE these examples. It exists purely for understanding what is needed to produce for each task)

- i. **Task:** List all Majors in the Major Dimension table in reverse order in a view called *v_List_Majors*
- ii. **SQL Statement:** "CREATE VIEW v_List_Majors AS SELECT major_description FROM dim_majors ORDER BY major_description DESC;"
- iii. **Task:** List total visits, total hours, and average time by Class (Freshman, Sophomore, Junior, Senior etc..) in a view called *v_Class*
- iv. **SQL Statement:** "CREATE VIEW V_CLASS AS SELECT DISTINCT CLS.CLASS_DESCRIPTION, SUM(TOTAL_VISITS) AS TOT_VISITS, SUM(TOTAL_TIME) AS TOT_MINUTES, SUM(TOTAL_TIME)/60 AS TOT_HOURS, (SUM(TOTAL_TIME)/60)/SUM(TOTAL_VISITS) AS AVG_TIME FROM USEFACT AS UF, DIM_MAJOR AS MJ, DIM_CLASS AS CLS, DIM_GROUP_PROJECTS AS GP

```
WHERE UF.GROUP_PROJECT_ID = GP.GROUP_PROJECT_ID AND
UF.MAJOR_CODE = MJ.MAJOR_CODE AND
UF.CLASS_ID = CLS.CLASS_ID
GROUP BY CLS.CLASS_DESCRIPTION;"
```

3. I will not lie :) This section **WILL** be difficult at first. It requires you to understand the relationships between the entities and between the **Production** and **LABINFO** data warehouse.

Submission

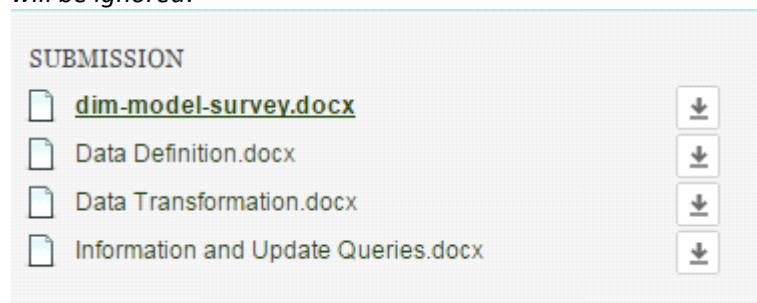
1. Use the appropriate section title (Data Definition, Data Injection etc...) at the top of each page
2. Add a page number and add your initials to bottom right corner of each page
3. Scan or take a picture of each page
4. Paste each image into a word document on its own page. Make sure the image is large enough to make it clear to the reader. If I can't read your writing, I will not give you credit.
5. Add the Title page to the **top** of the document using the following definition
 - a. **Section Title**
 - b. **Assignment Title**
 - c. **Student Name**
 - d. **Course and Section**
 - e. **Date and time the drawing was completed**
 - f. **Version in the form of v1, v2, v3 etc...**
6. The file name of the document should be the same as the section title
7. Upload **each** MS Word document to the assignment on BB (total of 4 Files)
8. Upload your last-revised/graded ERD from the assignment in MS Word or PowerPoint format
"Construct a Dimensional Model"

Submission Checklist

- ☐ 1. Everything is handwritten
- ☐ 2. All images are clearly visible
- ☐ 3. Every document has a title page, and page numbers/initials on each page
- ☐ 4. Documents are named properly
- ☐ 5. Upload 4 properly named MS Office Documents ([File 1](#), [File 2](#), [File 3](#), [ERD](#))

IMPORTANT:

When fully submitted your submission in BB should appear similar to the following example... **Zero Credit** will be given until your submission is properly submitted. 1 File is not a submission. *Ziped files will be ignored.*



FINAL NOTE:

I DON'T EXPECT this work product to be *completely finished* by the due date of this module. There **WILL** BE several revisions (as you did with the Dimensional Model Diagram). If you get stuck, contact me or come to the Lab Hours. But **PLEASE** do your part to understand what you are asking me. If you come to me or email me with the blanket statement "I don't know what to do" then I suggest you read the instructions again. Or ask a question about the first instruction that does not make sense to you. Many

of you want to be hand-fed a base-line understanding of what to produce without reading fully this ENTIRE assignment instruction set. The instructions are clear for students who really understand the underlying principles brought up by the material they have studied so far. I suggest you re-visit the material, ask a peer, or a subject-area specialist when it does not make sense. While I am happy to answer most questions (and many of you know I am), I DO expect you to do the requisite background work.

TURN IN *SOMETHING USEFUL* by the due date! You will NOT get credit for later revisions on late work. What I mean by useful is at the very least a complete DATA DEFINITION and almost complete DATA TRANSFORMATION. I will not say more about this. I know you want definite deliverable definitions. But I would rather you simply focus on completing the entire project phase :)

There will be a **grading rubric** applied so that you can see how your grade will be influenced by various assessment factors and ultimately understand your mistakes--after which you may revise and resubmit for a better grade. The rubric is indicated by a box at the top of the blackboard assignment.



Rubric Example Icon

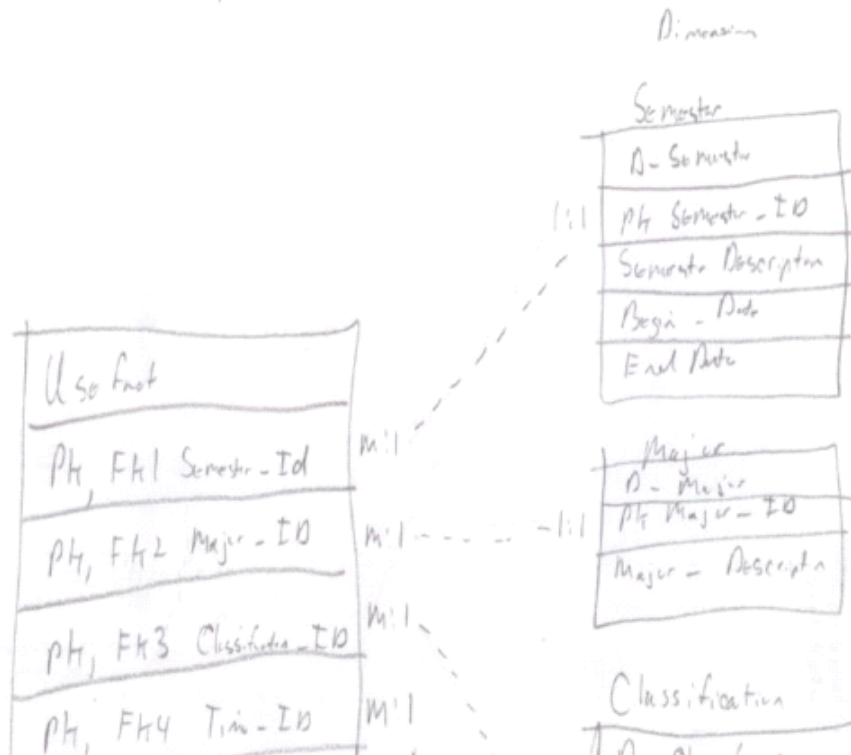
Subsequent attempts to LATE submissions will be ignored. Submit version 1 on or BEFORE the content area deadline.

ERD Example

Thursday, February 23, 2017

10:37 AM

Complete Model Diagram



Data Definition Example File

Thursday, February 23, 2017

10:37 AM

Data Definition

Project: Build a Data Warehouse - Phase 1

[REDACTED]

U 421 - 01W

11-23-2016 9:00 PM

V1 Revised

Data Definition

```
=> CREATE DATABASE IF NOT EXISTS 'LABINFO';  
=> CREATE TABLE 'LABINFO'.DIM_SEMESTER ('SEMESTER-  
ID VARCHAR(4) NOT NULL, 'SEMESTER_DESCRIPTION'  
VARCHAR(14), 'BEGIN-DATE' DATE, 'END-DATE' DATE,  
PRIMARY KEY ('SEMESTER-ID'));  
=> CREATE TABLE 'LABINFO'.DIM_MAJOR ('MAJOR-CODE'  
VARCHAR(4) NOT NULL, 'MAJOR-NAME' VARCHAR(35),  
PRIMARY KEY ('MAJOR-CODE'));  
=> CREATE TABLE 'LABINFO'.DIM_CLASS ('CLASS-ID'
```

Data Transformation Example File

Thursday, February 23, 2017 10:38 AM

DATA TRANSFORMATION

Project: Build a Data Warehouse - Phase 1

[REDACTED]

U421-01W

11-23-2016 7:00 PM

V1 - REVISED

DATA Transformation

Project: Build a Data Warehouse - Phase 1

U421-01W

11-23-2016

7:00 PM

V1

INSERT INTO DIM-SEMESTER (SEMESTER_ID, SEMESTER_DESCRIPTION,
BEGIN-DATE, END-DATE)

SELECT DISTINCT

CONCAT(

CASE WHEN MONTH (DATE) BETWEEN 1 AND 6 THEN 'SP'
WHEN MONTH (DATE) BETWEEN 7 AND 12 THEN 'FA'

END

, CONVERT (YEAR (DATE) - 2000, CHAR (2))) AS SEMESTER_ID

CONCAT(

CASE WHEN MONTH (DATE) BETWEEN 1 AND 6 THEN 'SPRING'
WHEN MONTH (DATE) BETWEEN 7 AND 12 THEN 'FALL'

END

, CONVERT (YEAR (DATE), CHAR (4))) AS SEMESTER_DESCRIPTION,

(CASE WHEN MONTH (DATE) BETWEEN 1 AND 6 THEN

Informational/Update Example File

Thursday, February 23, 2017

10:43 AM

INFORMATION AND UPDATE Query
PROJECT: Build a Data Warehouse - Phase 1

[REDACTED]
U421- GIW

11-23-2016 7:00 PM

V1 - REVISED

INFORMATION AND UPDATE Queries

1. a.)

i.) `SELECT * FROM 'LABINFO'. 'DIM-SEMESTER';`

(This Query returns the SEMESTER-ID, SEMESTER-DESCRIPTION, BEGIN-DATE and END-DATE for all rows of the DIM-SEMESTER table in the LABINFO DB)

`SELECT * FROM 'LABINFO'. 'DIM-CLASS';`

(This Query returns the CLASS-ID and CLASS-DESCRIPTION Columns for all rows of the DIM-CLASS table of the LABINFO DB)

ii) `SELECT 'SEMESTER-ID', 'END-DATE' FROM 'LABINFO'. 'DIM-SEMESTER';`

(This Query returns only SEMESTER-ID and END-DATE Columns for every row of the DIM-SEMESTER table of the LABINFO DB)

`SELECT 'MAJOR-CODE', 'CLASS-ID' FROM`

`'LABINFO'. 'DIM-CLASS';`

LABINFO Data Warehouse - Best Practices Model

Saturday, October 29, 2016 10:16 AM

Use the following guide if you are struggling to determining what SQL statements will be necessary to build the data warehouse - Please NOTE - Not all of the entities are represented. These are examples. You **MUST** alter something in ALL of them to make them fit **YOUR** ERD:

1. **Create ALL dimension table structures** (from your [ERD](#))
2. **Create fact table structure** (from your ERD)
3. **Create USELOG_ODS structure** - This temporary transformation table (Operational Data Store - ODS) is necessary because the schema of the operational table USELOG is optimized for use in running the lab. USELOG_ODS will be very similar to USELOG except needs to include all of the IDs like MAJOR_ID, SEMESTER_ID, TIME_ID in addition to the date, time, duration. See sample ERD. (See [Figure 1](#))
4. **Populate dimension table data for ALL Dimensions** - Dimension data should only include values that exist as a result of the USELOG table. *For example*- if there are 25 total Majors and only 14 Majors are represented in the USELOG table via its relationship to STUDENT, then only 14 Majors will be added to the DIM_MAJOR table. (clue: see figures 2 and 3)
5. **Populate USELOG_ODS**
 - a. Fill with atomic values that do not require logic to derive like duration and the IDs. TIME_ID and SEMESTER_ID have to be derived- they will be added in another step (See [Figure 4](#))
 - b. Update derived data... SEMESTER_ID AND TIME_ID (See [Figure 5](#))
6. **Populate USEFACT** with data from USELOG_ODS. Use Joins and aggregate functions to fill the duration and visit count data (See [Figure 6](#))
7. **Write queries on USEFACT** using Joins to dimension tables to produce useful information to be gathered for reporting. If you've gotten this far on your own, you can finish. Use the business rules from the "Construct a dimensional model" assignment to determine what SQL statements will be needed to present the boss with the desired decision data.

Figure 1

This is a transition table. Only necessary to stage the data for import into the USEFACT table - THERE ARE NO PKs in this table					
USELOG_ODS		Source			
SEMESTER_ID		derived - use and alter provided UPDATE_USELOG_ODS SQL Statement			
MAJOR_ID		from DIM_MAJOR			
CLASSIFICATION_ID		from DIM_CLASSIFICATION			
STUDENT_ID		from USELOG			
TIME_ID		derived - use and alter provided UPDATE_USELOG_ODS SQL Statement			
GROUP_PROJECT_ID		from DIM_GROUP_PROJECT			
DURATION		from USELOG			
DATE		from USELOG			
TIME		from USELOG			

Figure 2

```
INSERT INTO DIM_SEMESTER (<Fill in Your fields here!>)

SELECT DISTINCT
CONCAT (
    CASE WHEN MONTH (DATE) BETWEEN 1 AND 6 THEN 'SP'
    WHEN MONTH (DATE) BETWEEN 7 AND 12 THEN 'FA'
END
, CONVERT (YEAR (DATE) - 2000, CHAR (2))) AS SEMESTER_ID,
CONCAT (
    CASE WHEN MONTH (DATE) BETWEEN 1 AND 6 THEN 'SPRING '
    WHEN MONTH (DATE) BETWEEN 7 AND 12 THEN 'FALL '
END
, CONVERT (YEAR (DATE), CHAR (4))) AS SEMESTER_DESCRIPTION,

(
    CASE WHEN MONTH (DATE) BETWEEN 1 AND 6 THEN
STR_TO_DATE (CONCAT ("01/01/", CONVERT (YEAR (DATE),
CHAR (4))), "%m/%d/%Y")
    WHEN MONTH (DATE) BETWEEN 7 AND 12 THEN
STR_TO_DATE (CONCAT ("07/01/", CONVERT (YEAR (DATE),
CHAR (4))), "%m/%d/%Y")
END
) AS BEGINDATE,
(
    CASE WHEN MONTH (DATE) BETWEEN 1 AND 6 THEN
STR_TO_DATE (CONCAT ("06/30/", CONVERT (YEAR (DATE),
CHAR (4))), "%m/%d/%Y")
    WHEN MONTH (DATE) BETWEEN 7 AND 12 THEN
STR_TO_DATE (CONCAT ("12/31/", CONVERT (YEAR (DATE),
CHAR (4))), "%m/%d/%Y")
END
) AS ENDDATE

FROM USELOG;
```

```
INSERT INTO DIM_TIME (<Fill in Your fields here!>)
SELECT TM.TIME_ID, TM.TIME_DESCRIPTION, TM.BEGIN_TIME, TM.END_TIME
FROM (
SELECT DISTINCT
CASE
WHEN TIME (TIME) BETWEEN CAST ('06:01:00' AS TIME) AND CAST ('12:00:00'
AS TIME) THEN 1
WHEN TIME (TIME) BETWEEN CAST ('12:01:00' AS TIME) AND CAST ('18:00:00'
AS TIME) THEN 2
WHEN TIME (TIME) BETWEEN CAST ('18:01:00' AS TIME) AND CAST ('23:59:00'
AS TIME) OR
TIME (TIME) BETWEEN CAST ('00:00:00' AS TIME) AND CAST ('06:00:00' AS
TIME)
THEN 3
END AS TIME_ID,
CASE
WHEN TIME (TIME) BETWEEN CAST ('06:01:00' AS TIME) AND CAST ('12:00:00'
AS TIME) THEN 'MORNING'
WHEN TIME (TIME) BETWEEN CAST ('12:01:00' AS TIME) AND CAST ('18:00:00'
AS TIME) THEN 'AFTERNOON'
WHEN TIME (TIME) BETWEEN CAST ('18:01:00' AS TIME) AND CAST ('23:59:00'
AS TIME)
OR
TIME (TIME) BETWEEN CAST ('00:00:00' AS TIME) AND CAST ('06:00:00' AS
TIME)
THEN 'NIGHT'
END AS TIME_DESCRIPTION,
CASE
WHEN TIME (TIME) BETWEEN CAST ('06:01:00' AS TIME) AND CAST ('12:00:00'
AS TIME) THEN CAST ('06:01:00' AS TIME)
WHEN TIME (TIME) BETWEEN CAST ('12:01:00' AS TIME) AND CAST ('18:00:00'
AS TIME) THEN CAST ('12:01:00' AS TIME)
WHEN TIME (TIME) BETWEEN CAST ('18:01:00' AS TIME) AND CAST ('23:59:00'
AS TIME)
OR
TIME (TIME) BETWEEN CAST ('00:00:00' AS TIME) AND CAST ('06:00:00' AS
TIME)
THEN CAST ('18:01:00' AS TIME)
```

```

END AS BEGIN_TIME,
CASE
WHEN TIME(TIME) BETWEEN CAST('06:01:00' AS TIME) AND CAST('12:00:00'
AS TIME) THEN CAST('12:00:00' AS TIME)
WHEN TIME(TIME) BETWEEN CAST('12:01:00' AS TIME) AND CAST('18:00:00'
AS TIME) THEN CAST('18:00:00' AS TIME)
WHEN TIME(TIME) BETWEEN CAST('18:01:00' AS TIME) AND CAST('23:59:00'
AS TIME)
OR
TIME(TIME) BETWEEN CAST('00:00:00' AS TIME) AND CAST('06:00:00' AS
TIME)
THEN CAST('06:00:00' AS TIME)
END AS END_TIME
FROM PRODUCTION.USELOG) AS TM

```

Figure 3

```

INSERT INTO DIM_MAJOR (<Fill in Your fields here!>)
SELECT DISTINCT MJ.MAJOR_CODE, MJ.MAJOR_NAME
FROM STUDENT AS ST, MAJOR AS MJ
WHERE
ST.MAJOR_CODE = MJ.MAJOR_CODE;

```

```

WHERE
TM.TIME_ID IS NOT NULL AND TM.TIME_DESCRIPTION IS NOT NULL AND
TM.BEGIN_TIME IS NOT NULL AND TM.END_TIME IS NOT NULL
ORDER BY TM.TIME_ID;

```

Figure 4

```

INSERT INTO USELOG_ODS (<Fill in Your fields here!>)
SELECT
MJ.MAJOR_ID,
CLS.CLASS_ID,
STU.STUDENT_ID,
UL.DURATION,
GP.GROUP_PROJECT_ID,
UL.DATE,
UL.TIME

FROM
USELOG AS UL,
STUDENT AS STU,
DIM_GROUP_PROJECTS AS GP,
DIM_MAJOR AS MJ,
DIM_CLASS AS CLS

WHERE
UL.STUDENT_ID = STU.STUDENT_ID AND
UL.GROUP_PROJECT_ID = GP.GROUP_PROJECT_ID AND
STU.MAJOR_CODE = MJ.MAJOR_ID AND
STU.CLASS_ID = CLS.CLASS_ID;

```

Figure 5

```

UPDATE USELOG_ODS SET <add semester ID column> =
CONCAT(
CASE WHEN MONTH(DATE) BETWEEN 1 AND 6 THEN 'SP'
WHEN MONTH(DATE) BETWEEN 7 AND 12 THEN 'FA'
END
, CONVERT(YEAR(DATE) - 2000, CHAR(2))),

<add time ID column> =
CASE
WHEN TIME(TIME) BETWEEN CAST('06:01:00' AS TIME) AND CAST('12:00:00'
AS TIME) THEN 1
WHEN TIME(TIME) BETWEEN CAST('12:01:00' AS TIME) AND CAST('18:00:00'
AS TIME) THEN 2
WHEN
TIME(TIME) BETWEEN CAST('18:01:00' AS TIME) AND CAST('23:59:00' AS
TIME) OR
TIME(TIME) BETWEEN CAST('00:00:00' AS TIME) AND CAST('06:00:00' AS
TIME)
THEN 3
END;

```

Figure 6

```

INSERT INTO USEFACT ( SEMESTER_ID, MAJOR_CODE, CLASS_ID,
TIME_ID, GROUP_PROJECT_ID, TOTAL_VISITS, TOTAL_TIME )
SELECT DISTINCT SEMESTER_ID, MAJOR_CODE, CLASS_ID, TIME_ID,
GROUP_PROJECT_ID, <add aggregate function> AS TOTAL_VISITS, |
<add aggregate function> AS TOTAL_MINUTES
FROM USELOG_ODS
GROUP BY <add columns here>;

```