

Data Warehouse Implementation for the University of the Mountains Project Plan

Aaron Jones, Badou Kossonou, Hanif Lumsden

University of Maryland Global Campus

DBST 665 9040 Data Warehouse Technologies

Dr. Kuchibhotla

07/26/2022

Table of Contents

Abstract	3
Introduction	4
Project Plan	4
Project Management	4
Requirement Analysis	7
Data Model	8
ETL Model and Method	10
Report Analysis	11
Problems and Issues	12
Conclusion	12
References	14
Appendix	15

Abstract

A data warehouse solution is proposed and implemented by Group 2 of DBST 665 for the University of the Mountains. Stakeholders at this university want a data solution where the data is centralized, readily accessible, and can answer their pressing questions. Analysis is needed for their course registration data for stakeholders to make business decisions with the aim of expanding the university's appeal and budget. Using Oracle SQL*Loader, SQL*Plus, and Oracle SQL Developer, data in the form of comma-separated-values undergoes the process of extract-transform-load (ETL) to populate data warehouse dimensions: student, professor, class, academic program; and the fact table as course registration. Upon creation, the data warehouse is used to answer the university's immediate inquiries as a demonstration of its insight capabilities. The data warehouse reveals the mean grade performance and enrollment statistics.

Keywords: Data warehouse, ETL, course registration, university, Oracle, grades, enrollment

Introduction

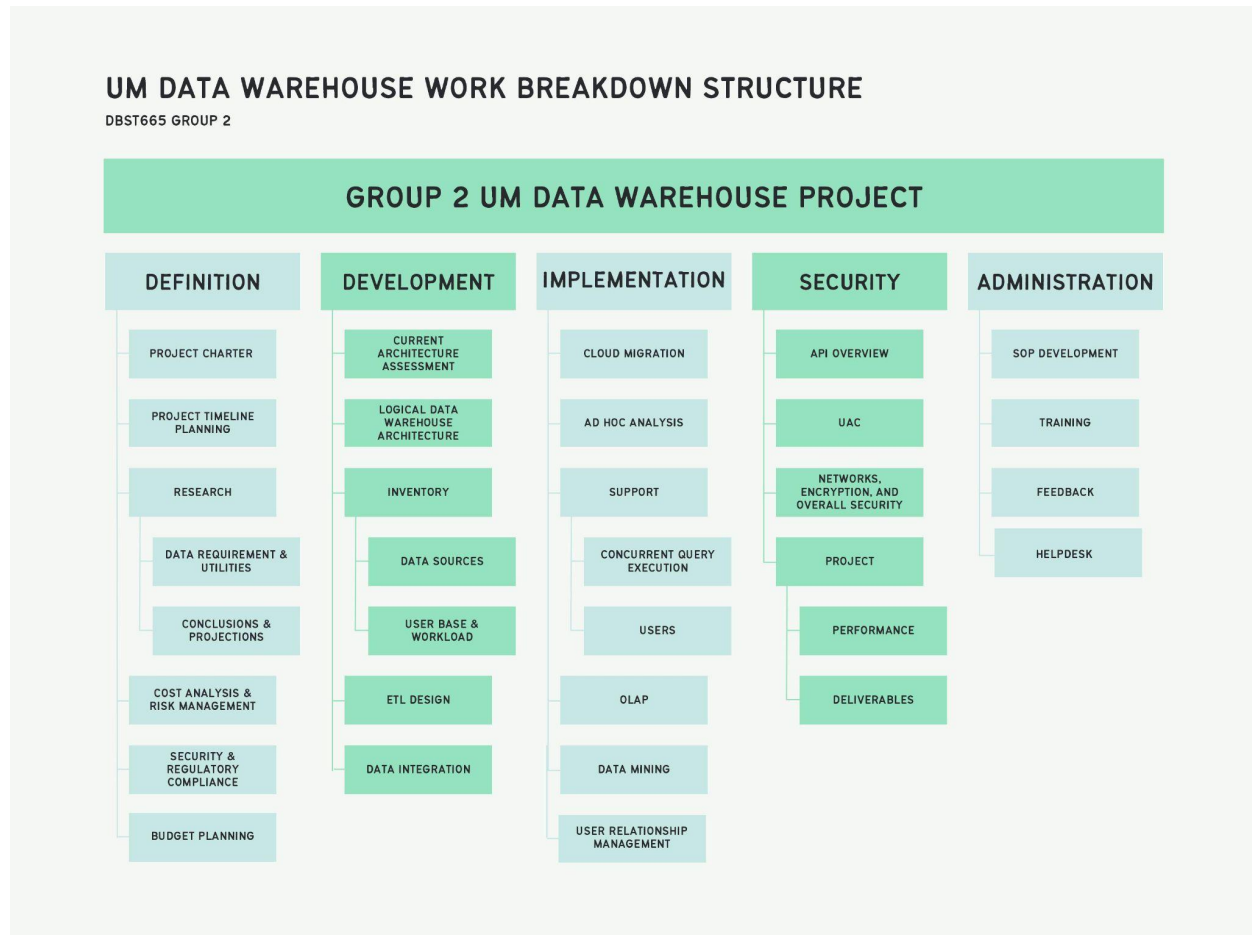
The University of the Mountains will utilize funding towards a fully optimized data warehouse solution for class registration to save money in the long-term. University management hires DBST665 Group 2 to fulfill this requirement. University of the Mountains's current operational data store is posing problems with data extraction in a timely manner. Users are running into messy data, inconsistency, and availability issues. This poses problems for research faculty and student experience inhibiting the university's task of their students reaching their educational goals. Analysts at the University of the Mountains come to unproductive conclusions due to the state of the data store. It is in the University's best interest to seek data warehouse implementation as the benefits it comes with will lead to better school performance overall and more funding in the long run.

This academic data warehouse solution will be an accessible repository of information extracted from transaction networks across the campus and act as the official source for university data. University stakeholders will be able to see the big picture that will enable informed decision making. Group 2 decides to model the Class registration fact table with four dimensions: student, class, professor, and academic program. Group 2 aims to create a data warehouse that will improve access efficiency and data reporting through well-documented data schematics. The data warehouse is expected to be implemented in approximately 11 months time according to the Gantt chart.

Project Plan

Project Management

Group 2 breaks down the project into smaller tasks for a more manageable approach. The scope and schedule is integrated in **figure 1**:

Figure 1*Work Breakdown Structure*

Note: Work Breakdown structure created using the Canva web application.

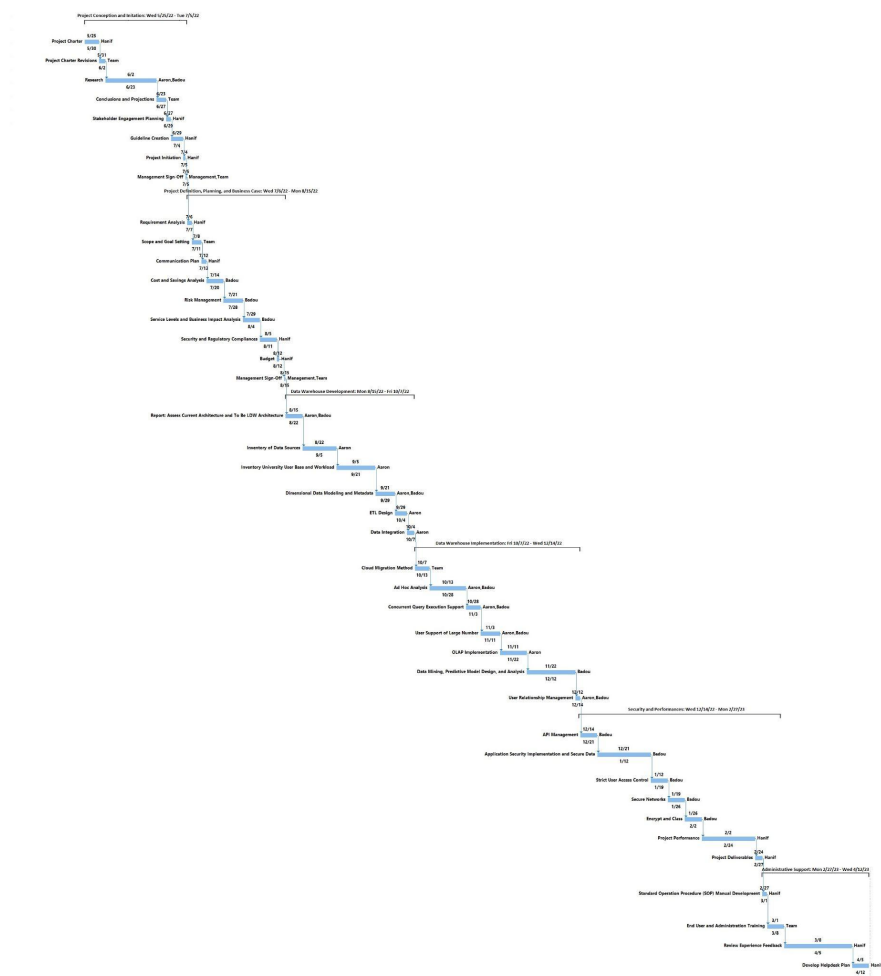
The project team is composed of three people: Aaron Jones, the developer and scribe who is responsible for developing the data warehouse and other information relating to business intelligence and writing notes; Badou Kossonou, the analyst that will research, aid in warehouse design and management, and evaluate data to make business recommendations; Hanif Lumsden, the team lead and project manager who will oversee the project.

According to the gantt chart, the project will last from approximately 11 months from May 25, 2022 to April 12, 2023 and consist of six phases: project conception and initiation;

project definition planning and business case; data warehouse development; data warehouse implementation; security and performance; lastly, the administrative support.

Figure 2

Gantt Chart



Note: Full resolution screenshot is [hosted here](#) on Google Drive.

As seen in **Figure 2**, the project starts with a project charter and revisions, where the project plan, scope and participants are written in a report. Research on potential stakeholders, requirements, necessary tools, and data warehouse implementation is done for three weeks followed by conclusions and stakeholder engagement planning that will inform the entire project

guideline. Post-project initiation, the team will undergo a month-long review regarding project analysis, risk management, business impact and compliance.

A report is written regarding the current data structure design and the to-be logical data warehouse architecture. Data sources, user base and workload will undergo inventory. After inventory, dimensional data modeling and metadata development is underway; subsequently, ETL design followed by data integration commences. Data warehouse implementation will finally go underway, utilizing Oracle SQL Developer and SQL*Loader.

Group 2 will choose a cloud migration method. Afterwards, ad hoc analysis is done to help stakeholders design business needs moving forward. Developers and analysts will initiate concurrent query execution support, ensure data warehouse support for a large number of users, implement the online analytical processing server and data mining models that meet business requirements, followed by user relationship management. Stakeholders are worked with to assess and execute API management and security where the networks are secured and data is encrypted. A strict user access control is done granting certain administrative privileges to senior users. After the data warehouse creation and security fulfillment, project performance is assessed and deliverables are completed. University of the Mountain staff will undergo training according to the standard operation procedure, give experience feedback, and help form a helpdesk plan.

Requirement Analysis

The data source is from the university's transactional registration servers where the student's registration is processed. A data warehouse will function to handle the processes, reports, and analysis. Undergraduates, graduates, faculty, and enrollment information are all added to the data warehouse and keeps track of the 'student lifecycle' (Boston University, 2022). Decision-making and strategic tasks are enabled through this warehouse. University of the

Mountain stakeholders want immediate inquiries about overall GPA, marketing target base, and professor's teaching effectiveness on their students' performance to test data warehouse capabilities.

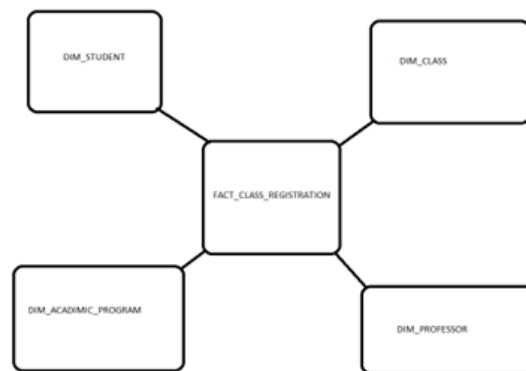
Data Model

Conceptual

Concept model on **figure 3** represents the data warehouse system in simplified, star schema form. The dimension consists of student, class, registration, and professor. The dimensions contain attributes of the fact table class registration.

Figure 3

Conceptual Model Design

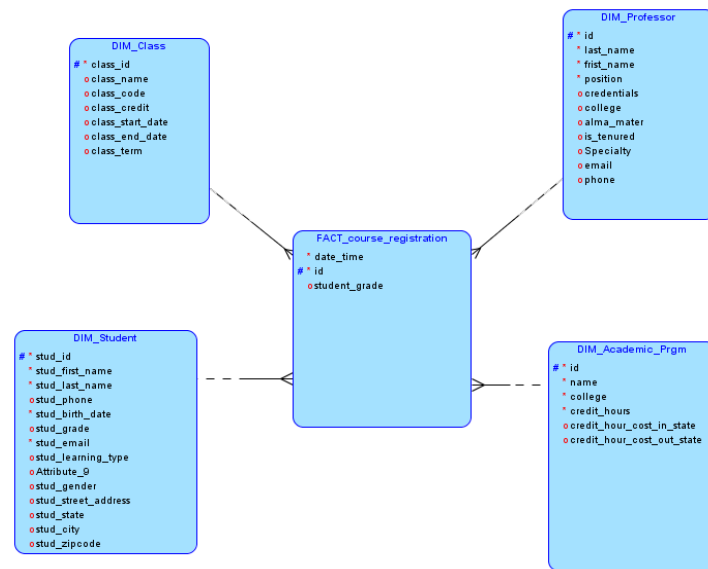


Logical

Oracle SQL Developer Data Modeler is used to generate the logical model below and the physical model in the section after that. A single view is provided of the data warehouse, validating business requirements and relationships. All attributes are displayed in the fact and dimension table in the logical model with the primary key displayed in each one.

Figure 4

Logical Model Design



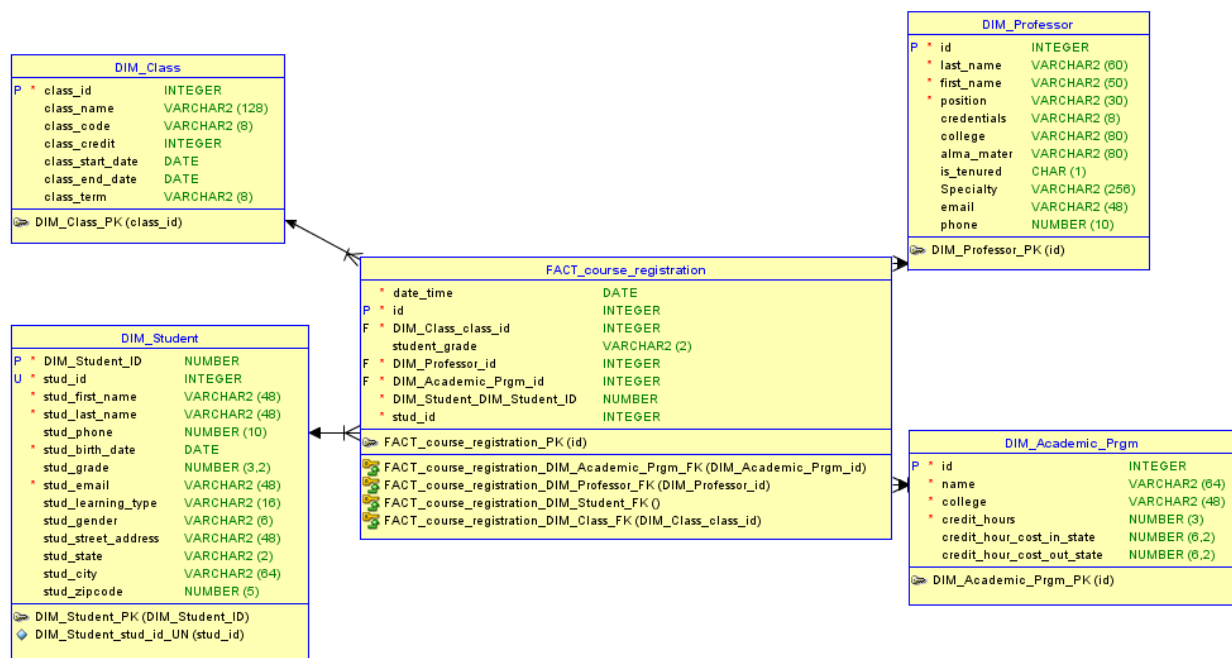
Note: Full resolution screenshot is [hosted here](#) on Google Drive.

Physical

The physical model lists the complete integrity constraints and data type.

Figure 5

Physical Model Design



Note: Full resolution screenshot is [hosted here](#) on Google Drive.

The business rules are as follows:

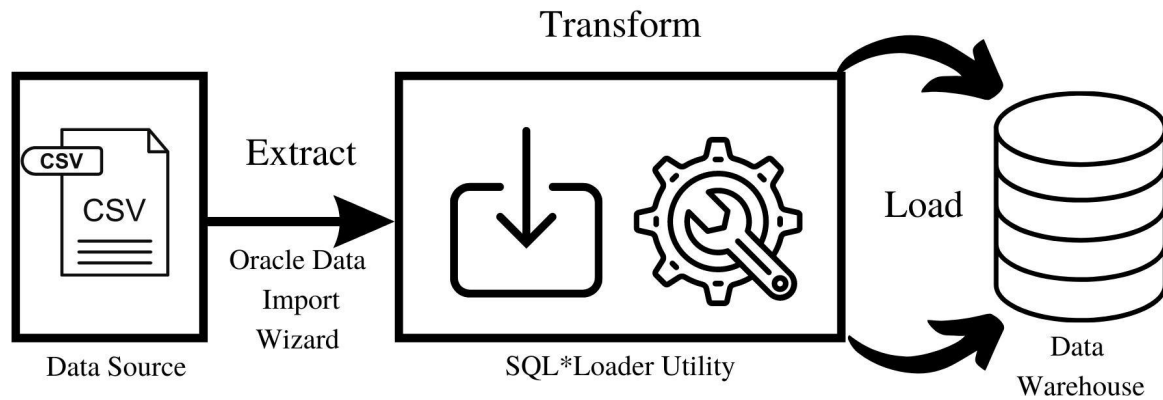
1. A CLASS contains one or many COURSE REGISTRATIONS.
2. A COURSE REGISTRATION belongs to a single CLASS.
3. A STUDENT is registered for one or many COURSE REGISTRATIONS.
4. One COURSE REGISTRATION record belongs to one STUDENT.
5. A PROFESSOR is registered for one or many COURSE REGISTRATIONS.
6. One COURSE REGISTRATION record belongs to one professor.
7. An ACADEMIC PROGRAM contains one or many COURSE REGISTRATIONS.
8. A COURSE REGISTRATION record belongs to one ACADEMIC PROGRAM.

ETL Model and Method

This step identifies data sources, proces, and method to load the data. The complete ETL script is in the appendix section. Dimension tables are in comma-separated-value (.csv) format. Using Oracle Data Import Wizard, the data is imported and the create table script is generated along with the control, shell, and batch file for each dimension. For the fact table, a fact extract, good and bad table is created to partition the useful and non-useful fact data into the final course registration fact table. Data is loaded into the dimensions and fact table using the windows terminal: initializing SQL*Loader, logging in, and loading the respective control files. Afterwards, the data is transformed and both the good and bad tables are populated using SQL*Plus. Data is then loaded into the registration fact table using the MERGE function.

Figure 6

ETL Flow Chart



Report Analysis

University of the Mountains seeks insight on the registration data. The university needs four immediate insights regarding the average grade performance from students for each class to see what degree the material is reaching students; undergraduate and graduate enrollment total as well as enrollment total through the years is inquired about; lastly class performance by professor to evaluate professor performance. The following questions are requested to be answered using the newly installed data warehouse and the queries can be referred to in the appendix: overall mean grade performance from students for each class, total undergraduate and graduate enrollment, total registration report over time, and each professor's class performance.

Queries show average grade performance for each class is lower than 90, but more than 87. Classes generally have students getting a B+, or ending the semester with a GPA of 3.3 assigned to their cumulative GPA. Average grade performance results highlight a problem. This means the University of the Mountains should re-evaluate their material, teaching style, accessibility, and many other factors with the aim of increasing the overall grade performance. For undergraduate and graduate enrollment total, the majority of enrollment comes from graduate students. The marketing and outreach team should consider that graduate student prospects are their target base when generating advertisements both physical and digital.

Registration total has risen greatly in the first 4 years with the total registration rate gradually lowering since 2021. Another consideration of the marketing and outreach team is to evaluate why registration rate has decreased amid the coronavirus outbreak. University of the Mountains should evaluate their online platform and present it more during advertising. Each professor's performance is evaluated, with four professor's course performance averaging more than the average grade performance as a whole and five underperforming; the rest are within average grade performance range. This insight shows that the five underperforming professors might do better to teach other types of material.

Problems and Issues

The date format for the dim-class.csv is not uniform. All dates in the dim-class.csv need to follow the DD-MM-RR to avoid formatting errors and should be fixed manually. Using the Oracle Data Import Wizard and insert SQL*Loader Utility method, all data types need to be adjusted mirroring the ETL script in the appendix section. The fact_table_source.csv contains NULL values and wrong data format for some portions. A method followed to remedy the .csv file is to delete the rows containing NULL values and changing the '3o' to 30 in the using the search function in a text editor. For the 'Datetime' attribute in the fact table, the date should follow the 'mm/dd/yyyy HH24:MI' format.

Conclusion

A data warehouse solution is implemented for the University of the Mountains. It is in the best interest of the stakeholders to use Group 2's data warehouse for analysis of the course registration. Analysis specifically on grade performance and enrollment. Data models and the ETL script are created. It is reported that enrollment rate for the University of the Mountain increased at a steady rate of more than 100 each year since 2017, but slowed down since 2021.

Graduate students compose most of the student population. The average grade for each class is a B+ and the professors are ranked based on the performance of their classes.

With the data warehouse solution installed, data requirement analysts are capable of answering these questions and more. Conclusions will lead to better business decisions and marketing plans, which is always a benefit of data warehouses when implemented properly. With registration data centralized, answers to the university's requirements are able to be queried faster. Transactions in registrations are managed and organized to be more useful. This is the beginning of a newer business intelligence for the University of the Mountains. As this is a newer solution, help desk staff will be in charge of training and onboarding the end users on the new system and scope decisions.

References

Boston University. (2022). *Academic data warehouse (ADW)*. Boston University: Information Services and Technology.

<https://www.bu.edu/tech/services/data-management-analytics/academic-reporting-analytics/student-reporting-analytics/adw-resources/>

Appendix

ETL Script

The following is the ETL Script used for the project. CSV files are premade and data from them is imported into SQL Developer using SQL Loader. Command Prompt is opened using windows key + r and typing in 'cmd' \ and the working directory is set to the desktop:

```
cd c:\users\studentfirst\desktop
```

SQL Plus is initialized in the shell:

```
sqlplus / as sysdba
Enter user-name: DBST_USER
Enter password: SecurePassword
```

Table is created using script generated from Oracle Data Import Wizard using the [class-dim.csv](#), [DIM_ACADEMIC_PRGM.csv](#), [DIM_PROFESSOR.csv](#), and [student_dim.csv](#) and choosing import method 'SQL*Loader Utility'. Datatypes are adjusted and matched as below and constraints are added. Code is ran in SQL Plus (Oracle SQL Worksheet is okay too):

*Note, fix the date format for all date entries in the dim-class.csv to DD-MM-RR uniformly manually.

```
SET DEFINE OFF
CREATE TABLE DIM_CLASS (
  class_id INTEGER NOT NULL,
  class_name VARCHAR2(128),
  class_code VARCHAR2(128),
  class_credit INTEGER,
  class_start_date DATE,
  class_end_date DATE,
  class_term VARCHAR2(26));

ALTER TABLE DIM_CLASS ADD CONSTRAINT CLASS_pk PRIMARY KEY ( class_id );
```

```

SET DEFINE OFF
CREATE TABLE DIM_ACADEMIC_PROGRAM (
ID INTEGER NOT NULL,
NAME VARCHAR2(128) NOT NULL,
COLLEGE VARCHAR2(128) NOT NULL,
TYPE VARCHAR2(26),
CREDIT_HOUR_COST_IN_STATE NUMBER(6, 2),
CREDIT_HOUR_COST_OUT_STATE NUMBER(6, 2),
CREDIT_HOURS NUMBER(3) NOT NULL);

ALTER TABLE DIM_ACADEMIC_PROGRAM ADD CONSTRAINT AC_PGRM_pk PRIMARY KEY ( id
);

SET DEFINE OFF
CREATE TABLE DIM_PROFESSOR (
ID INTEGER NOT NULL,
LAST_NAME VARCHAR2(60) NOT NULL,
FIRST_NAME VARCHAR2(50) NOT NULL,
POSITION VARCHAR2(30) NOT NULL,
CREDENTIAL VARCHAR2(26),
COLLEGE VARCHAR2(80),
ALMA_MATER VARCHAR2(128),
IS_TENURED CHAR(3),
SPECIALTY VARCHAR2(512),
EMAIL VARCHAR2(48),
PHONE VARCHAR2(15));

ALTER TABLE DIM_PROFESSOR ADD CONSTRAINT PROF_pk PRIMARY KEY ( id );

SET DEFINE OFF
CREATE TABLE DIM_STUDENTS (
stud_id INTEGER NOT NULL,
stud_first_name VARCHAR2(48) NOT NULL,
stud_last_name VARCHAR2(48) NOT NULL,
stud_phone VARCHAR2(15),
stud_birth_date DATE NOT NULL,
stud_grade NUMBER(5, 3),
stud_email VARCHAR2(48) NOT NULL,
stud_learning_type VARCHAR2(16),
stud_gender VARCHAR2(10),
stud_street_address VARCHAR2(128),
stud_state VARCHAR2(2),
stud_city VARCHAR2(64),

```



```

stud_zipcode NUMBER(5));

ALTER TABLE DIM_STUDENTS ADD CONSTRAINT STUDENT_pk PRIMARY KEY ( stud_id );

/* .ctl .sh and .bat files are found in C:\Users\StudentFirst\Desktop. */

```

SQL Plus quit its process using this command within the command prompt:

```
exit
```

The following is generated into the control file:

```

/* .ctl file code for CLASS DIM */
load data
infile 'class-dim.csv' "str '\n'"
append
into table DIM_CLASS
fields terminated by ','
OPTIONALLY ENCLOSED BY '"' AND '"'
trailing nullcols
( class_id,
  class_name CHAR(4000),
  class_code CHAR(4000),
  class_credit,
  class_start_date DATE "DD-MON-RR",
  class_end_date DATE "DD-MON-RR",
  class_term CHAR(4000)
)
/* .ctl file code for ACADEMIC PROGRAM DIM */
load data
infile 'DIM_ACADEMIC_PRGM.csv' "str '\r\n'"
append
into table DIM_ACADEMIC_PROGRAM
fields terminated by ','
OPTIONALLY ENCLOSED BY '"' AND '"'
trailing nullcols
( ID,
  NAME CHAR(4000),
  COLLEGE CHAR(4000),

```

```

        TYPE CHAR(4000),
        CREDIT_HOUR_COST_IN_STATE,
        CREDIT_HOUR_COST_OUT_STATE,
        CREDIT_HOURS
    )

/* .ctl file code for PROFESSOR DIM */
load data
infile 'DIM_PROFESSOR.csv' "str '\r\n'"
append
into table DIM_PROFESSOR
fields terminated by ','
OPTIONALLY ENCLOSED BY '"' AND '"'
trailing nullcols
( ID,
  LAST_NAME CHAR(4000),
  FIRST_NAME CHAR(4000),
  POSITION CHAR(4000),
  CREDENTIAL CHAR(4000),
  COLLEGE CHAR(4000),
  ALMA_MATER CHAR(4000),
  IS_TENURED CHAR(4000),
  SPECIALTY CHAR(4000),
  EMAIL CHAR(4000),
  PHONE CHAR(4000)
)

/* .ctl file code for STUDENT DIM */
load data
infile 'student_dim.csv' "str '\r\n'"
append
into table DIM_STUDENT
fields terminated by ','
OPTIONALLY ENCLOSED BY '"' AND '"'
trailing nullcols
( stud_id,
  stud_first_name CHAR(4000),
  stud_last_name CHAR(4000),
  stud_phone CHAR(4000),
  stud_birth_date DATE "DD-MON-RR",
  stud_grade,
  stud_email CHAR(4000),
  stud_learning_type CHAR(4000),

```

```

stud_gender CHAR(4000),
stud_street_address CHAR(4000),
stud_state CHAR(4000),
stud_city CHAR(4000),
stud_zipcode
)

```

SQL Loader Utility is invoked within the command prompt:

```

sqlldr userid=DBST_USER control=class-dim.ctl log=track1.log
Password: SecurePassword
sqlldr userid=DBST_USER control=DIM_ACADEMIC_PRGM.ctl log=track2.log
Password: SecurePassword
sqlldr userid=DBST_USER control=DIM_PROFESSOR.ctl log=track3.log
Password: SecurePassword
sqlldr userid=DBST_USER control=student_dim.ctl log=track4.log
Password: SecurePassword

```

Data is then loaded into the Oracle SQL Developer.

FACT TABLE ETL SCRIPT

The process for the FACT table is below. First, the tables are created. For FACT_EXTRACT, the script is generated using Oracle Data Import Wizard with the FACT TABLE SOURCE. Refer to the issues portion of this paper (delete the rows with NULL and change the 3o to 30 in the [FACT_TABLE_SOURCE.csv](#) file):

*Note: change the datatype to the respective types listed below

```

SET DEFINE OFF /*import the FACT_TABLE_SOURCE.csv*/
CREATE TABLE FACT_EXTRACT (
id INTEGER,
student_ID INTEGER,
course_id INTEGER,
Class_name VARCHAR2(128),
letter_grade VARCHAR2(26),
grade NUMBER(38),
datetime DATE, /*change to mm/dd/yyyy HH24:MI format*/
semester VARCHAR2(26),

```

```

professor_id INTEGER,
Professor_name VARCHAR2(26),
Academic_PRGM_Degree_ID INTEGER);

/* .ctl .sh and .bat files are found in C:\Users\StudentFirst\Desktop. */

CREATE TABLE FACT_GOOD (
  Id INTEGER,
  student_id INTEGER,
  course_id INTEGER,
  Professor_id INTEGER,
  Academic_Prgm_Degree_id INTEGER,
  Datetime DATE,
  Grade NUMBER(38));

CREATE TABLE FACT_BAD (
  Class_name VARCHAR2(128),
  letter_grade VARCHAR2(26),
  semester VARCHAR2(26),
  professor_name VARCHAR2(26));

CREATE TABLE FACT_COURSE_REGISTRATION (
  Id INTEGER NOT NULL,
  student_id INTEGER NOT NULL,
  course_id INTEGER NOT NULL,
  Professor_id INTEGER NOT NULL,
  Academic_Prgm_Degree_id INTEGER NOT NULL,
  Datetime DATE NOT NULL,
  Grade NUMBER(5,0));

ALTER TABLE FACT_COURSE_REGISTRATION ADD CONSTRAINT REG_pk PRIMARY KEY ( id
);
ALTER TABLE FACT_COURSE_REGISTRATION
  ADD CONSTRAINT REG_ACAD_PRGM_fk FOREIGN KEY ( Academic_prgm_degree_id
)
  REFERENCES DIM_ACADEMIC_PROGRAM ( id );
ALTER TABLE FACT_COURSE_REGISTRATION
  ADD CONSTRAINT REG_CLASS_fk FOREIGN KEY ( course_id )
  REFERENCES DIM_CLASS ( class_id );
ALTER TABLE FACT_COURSE_REGISTRATION
  ADD CONSTRAINT REG_PROF_fk FOREIGN KEY ( Professor_id )
  REFERENCES DIM_PROFESSOR ( id );
ALTER TABLE FACT_COURSE_REGISTRATION

```

```
ADD CONSTRAINT REG_ST_fk FOREIGN KEY ( student_id )
REFERENCES DIM_STUDENT ( stud_id );
```

Data is loaded into fact extract:

```
load data
infile 'FACT_TABLE_SOURCE.csv' "str '\r\n'"
append
into table FACT_EXTRACT
fields terminated by ','
OPTIONALLY ENCLOSED BY '"' AND '"'
trailing nullcols
( id,
  student_ID,
  course_id,
  Class_name CHAR(4000),
  letter_grade CHAR(4000),
  grade,
  datetime DATE "mm/dd/yyyy HH24:MI",
  semester CHAR(4000),
  professor_id,
  Professor_name CHAR(4000),
  Academic_PRGM_Degree_ID
)
```

Using in the cmd prompt:

```
sqlldr userid=DBST_USER control=FACT_TABLE_SOURCE.ctl log=track4.log
Password: SecurePassword
```

SQL Plus is initialized:

```
sqlplus / as sysdba
```

This is the transform step. Both tables are populated using:

```
INSERT ALL
```

```

INTO FACT_GOOD (
    id, student_id, course_id, professor_id, academic_prgm_degree_id,
    datetime, grade
) values (
    id, student_id, course_id, professor_id, academic_prgm_degree_id,
    datetime, grade
)
INTO FACT_BAD (
    class_name, letter_grade, semester, professor_name
) values (class_name, letter_grade, semester, professor_name
)
SELECT * FROM FACT_EXTRACT;

```

This is the load step. Data is loaded into registration fact:

```

MERGE INTO FACT_REGISTRATION FACT
USING (SELECT * FROM FACT_GOOD) GOOD
ON (
    FACT.id = GOOD.id AND
    FACT.student_id = GOOD.student_id AND
    FACT.course_id = GOOD.course_id AND
    FACT.professor_id = GOOD.professor_id AND
    FACT.academic_prgm_degree_id = GOOD.academic_prgm_degree_id)
WHEN MATCHED THEN UPDATE SET
    FACT.datetime = GOOD.datetime,
    FACT.grade = GOOD.grade
WHEN NOT MATCHED THEN INSERT VALUES
    (
        GOOD.id,
        GOOD.student_id,
        GOOD.course_id,
        GOOD.professor_id,
        GOOD.academic_prgm_degree_id,
        GOOD.datetime,
        GOOD.grade
    );

```

Questions

```
/*Average grade performance from students for each class?*/
```

```
SELECT
    c.class_name,
    c.class_code,
    CAST(
        AVG(
            f.grade
        ) AS DECIMAL(5,2)
    )
    AS GRADE_AVG
FROM FACT_COURSE_REGISTRATION f
    JOIN DIM_CLASS c
    ON c.class_id = f.course_id
GROUP BY
    c.class_name,
    c.class_code
ORDER BY
    GRADE_AVG
DESC;
```

Script Output x Query Result x		
All Rows Fetched: 13 in 0.007 seconds		
CLASS_NAME	CLASS_CODE	GRADE_AVG
1 DBST 670	Database Systems Administration	89.66
2 ITEC 630	Information Systems Analysis, Modeling, and Design	89.61
3 DBST 651	Relational Database Systems	89.08
4 UCSP 615	Orientation to Graduate Studies at UMGC	89.05
5 DBST 660	Advanced Data Modeling	88.45
6 DBST 663	Distributed Database Management Systems	88.36
7 DBS 652	Advanced Relational/Object-Relational Database Systems	88.25
8 ITEC 626	Information Systems Infrastructure	88.22
9 ITEC 640	Information Technology Project Management	88.05
10 ITEC 630	Advanced Data Modeling	88.03
11 DBST 660	Database Security	88
12 DBST 667	Data Mining	87.85
13 ITEC 625	Computer Systems Architecture	87.21

```
/*Colleges with the greatest amounts of undergraduate enrollment and
colleges with the greatest amounts of post-undergraduate enrollment*/
SELECT
    a.college,
    SUM(CASE
        WHEN a.type = 'Undergraduate' THEN 1 ELSE 0 END) AS UNDERGRAD_TOTAL,
    SUM(CASE
```

```

        WHEN a.type = 'Masters' OR a.type = 'Doctorate' THEN 1 ELSE 0 END) AS
        GRAD_TOTAL
FROM
    DIM_ACADEMIC_PROGRAM a
    JOIN FACT_COURSE_REGISTRATION f
    ON a.id = f.academic_prgm_degree_id
GROUP BY
    a.college;

```

Worksheet Query Builder

```

SELECT
    a.college,
    SUM(CASE
        WHEN a.type = 'Undergraduate' THEN 1 ELSE 0 END) AS UNDERGRAD_TOTAL,
    SUM(CASE
        WHEN a.type = 'Masters' OR a.type = 'Doctorate' THEN 1 ELSE 0 END) AS
        GRAD_TOTAL
FROM
    DIM_ACADEMIC_PROGRAM a
    JOIN FACT_COURSE_REGISTRATION f
    ON a.id = f.academic_prgm_degree_id
GROUP BY
    a.college;

```

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0.026 seconds

COLLEGE	UNDERGRAD_TOTAL	GRAD_TOTAL
1 College of Information	47	330
2 College of Engineering	0	402

```

/*Total registration report over time*/
SELECT
    YEAR,
    SUM(N) OVER(
        ORDER BY
            YEAR
        ) TOTAL_REGISTRATION
FROM
    (
        SELECT
        EXTRACT(
            YEAR
            FROM
            DATETIME
        ) YEAR,

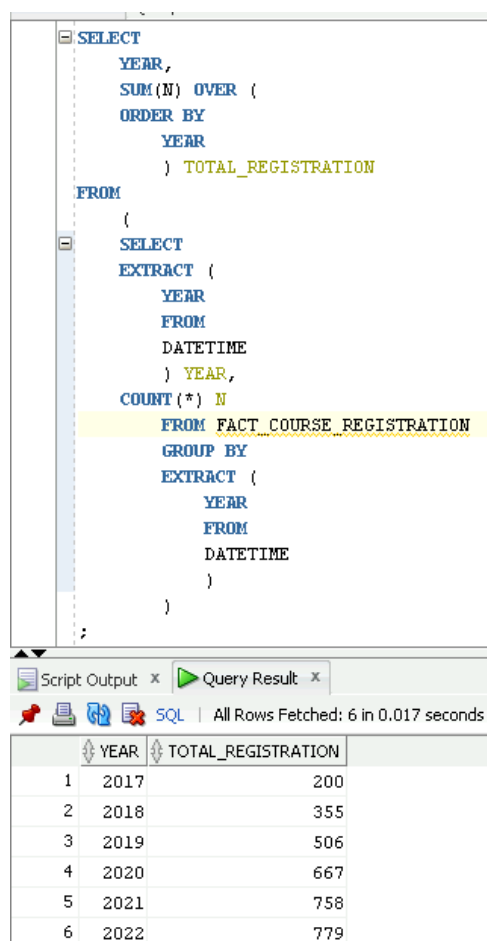
```



```

COUNT(*) N
FROM FACT_COURSE_REGISTRATION
GROUP BY
EXTRACT(
    YEAR
FROM
    DATETIME
)
)
;

```



The screenshot shows a SQL IDE with a query editor and a results pane. The query in the editor is a window function that calculates the total registration count per year. The results pane shows a table with two columns: YEAR and TOTAL_REGISTRATION, with data for the years 2017 through 2022.

```

SELECT
    YEAR,
    SUM(N) OVER (
        ORDER BY
            YEAR
    ) TOTAL_REGISTRATION
FROM
    (
        SELECT
            EXTRACT (
                YEAR
            FROM
                DATETIME
            ) YEAR,
            COUNT (*) N
        FROM FACT_COURSE_REGISTRATION
        GROUP BY
            EXTRACT (
                YEAR
            FROM
                DATETIME
            )
    )
;

```

Script Output x Query Result x

SQL | All Rows Fetched: 6 in 0.017 seconds

	YEAR	TOTAL_REGISTRATION
1	2017	200
2	2018	355
3	2019	506
4	2020	667
5	2021	758
6	2022	779

```

/*Professor's class performance*/
SELECT
    f.PROFESSOR_ID,
    p.FIRST_NAME,
    p.LAST_NAME,
    p.CREDENTIAL,
    p.IS_TENURED,

```

```
        CAST(  
        AVG(  
            f.grade  
        ) AS DECIMAL(5, 2)  
        )  
        AS GRADE_AVG  
FROM FACT_COURSE_REGISTRATION f  
    JOIN DIM_PROFESSOR p  
    ON p.id = f.professor_id  
GROUP BY  
    f.PROFESSOR_ID,  
    p.FIRST_NAME,  
    p.LAST_NAME,  
    p.CREDENTIAL,  
    p.IS_TENURED  
ORDER BY  
    GRADE_AVG  
    DESC;
```

Worksheet

Query Builder

```
SELECT
  f.DIM_PROFESSOR_ID,
  p.FIRST_NAME,
  p.LAST_NAME,
  p.CREDENTIAL,
  p.IS_TENURED,
  CAST (
    AVG (
```

Query Result x

SQL | All Rows Fetched: 22 in 0.078 seconds

	DIM_PROFESSOR_ID	FIRST_NAME	LAST_NAME	CREDENTIAL	IS_TENURED	GRADE_AVG
1	32	Hassan	Takabi	Ph.D.	No	92.67
2	30	Ian	Parberry	Ph.D.	No	91.05
3	33	Paul	Tarau	Ph.D.	No	90.59
4	29	JungHwan	Oh	Ph.D.	No	90.48
5	36	Hui	Zhao	Ph.D.	Yes	89.69
6	35	Xiaohui	Yuan	Ph.D.	No	89.4
7	31	Farhad	Shahrokhi	Ph.D.	Yes	89.22
8	34	Qing	Yang	Ph.D.	No	89.14
9	19	Song	Fu	Ph.D.	No	88.96
10	15	Bryce	Renee	Ph.D.	Yes	88.83
11	23	Krishna	Kavi	Ph.D.	No	88.54
12	24	Stephanie	Ludi	Ph.D.	Yes	88.45
13	16	Bill	Buckles	Ph.D.	Yes	87.94
14	18	Hyunsook	Do	Ph.D.	No	87.72
15	25	Armin	Mikler	Ph.D.	No	87.69
16	20	Xuan	Guo	Ph.D.	No	87.43
17	26	Saraju	Mohanty	Ph.D.	Yes	87.22
18	28	Rodney	Nielsen	Dual Ph.D.	Yes	87.04
19	17	Ram	Dantu	Ph.D.	Yes	87.03
20	27	Kirill	Morozov	Ph.D.	No	86.68
21	22	Wei	Jin	Ph.D.	No	86.47
22	21	Yan	Huang	Ph.D.	Yes	84.62