|  |  |  |
| --- | --- | --- |
| **StudentIDs:** |  | Score: |
| **Class:** |  |  |
| **Date:** |  |

This mini project assignment consists of 2 tasks. You should focus on providing complete solution, however if you are unable to solve a particular step, try to give at least a partial solution or provide justification for the reason for the lack of a solution.

# **Mini-project 2 – Final Analysis**

The next step of creating a dimensional data warehouse is the design of the dimensional model. Having specified the "business needs" and reality (problem field and data details) of the available data resources, we can start designing an adequate dimensional solution. In short, you should already understand the domain, know all the facts and contexts of their analysis (from stage 1). Further, using the dimensional model you can specify the blueprint for the ETL process. You need to specify the required data movement tasks (in the form of logical data map) and the plan of data handling (in the form of a basic ETL map).

The process of implementing an ETL system for a data warehouse system should be preceded by a deep and thorough understanding of the "business needs" and reality (problem field). A particular implementation of data movement and integration tasks can end up in a disaster if the goal is ill-defined or unclear. As such, here we focus on the analysis part and introduce a proper multidimensional model, i.e., model capable of handling all the identified user information needs.

## **Task 1.1 – Dimensional model – details:**

Please prepare the dimensional model for the selected dataset and specified user requirements, in accordance with the below specification and discuss the solution with the lecturer.

## **Task 1.1 – Dimensional model – Solutions**:

### 1.1 Dimensional Schema – Diagram

Please, define schema and prepare a diagram for multi-dimensional model (star, snowflake, or fact constellation). First, briefly present and explain your design solution for the fact table. Next, focus on presentation and explanation of dimension tables. Finally, design and present the final dimensional diagram.

**A screenshot of a computer

Description automatically generated**

## **Task 1.2 – ETL model – details:**

Please prepare the model of the ETL process, in accordance with the below specification and discuss the solution with the lecturer.

## **Task 1.2 – ETL model – Solutions**:

### 1.2 ETL map

Prepare an ETL map that specifies the required operations on data in the ETL process. Draft the map at a relatively high level of detail (focusing on dimension tables, rather than individual source-to-target mappings, as in logical data map), use a box to represent each utilised source table (file, spreadsheet, or other source), and a box to represent each dimensional table (table in the dimensional model). In short, the map should specify all operations, which are required to obtain the final dimensional table. Please, specify a map separately for each dimension table and separately for each fact table. Exemplar map is provided at the end of this document – in your submission, please remove them.

NOTE: basic example of an ETL map is provided at the end of this document.

…

# **Mini-project 2 – Implementation**

The focus of this stage is on implementation tasks: implementation of the ETL process using SQL Server Integration Services (SSIS) and implementation of a multidimensional cube (OLAP Cube) using SQL Server Analysis Services (SSAS). You will utilise the artifacts prepared in the previous stage, where you have managed to design a detailed multidimensional model schema (star/snowflake) and an overall ETL process map.

The draft of the map serves as the blueprint for the ETL process. Now your task is to implement the process. Bear in mind that due to limited resources (mostly time) we focus on a simplified task of implementing an exemplar ETL solution, rather than building a complete, more general, ETL system. You have already specified all needed data movement, cleaning, and conformation tasks. Now it is time to act. As such, in Task 1 you need to implement the planned pipeline using SSIS (if you prefer to use a different ETL tool, or set of tools, please consult your selection with the teacher – but generally it is possible and encouraged).

## **Task 2.1 – ETL PROCESS – details:**

Before the start, please divide your dataset into historical load data (data that is initially loaded into your data mart) and current load data batches (at least, two data batches that can be later loaded incrementally). Please prepare appropriate implementation of the historical load (remember to create needed target data structures) and incremental loads (remember to automate the entire process, keep metadata of your runs). Obviously, both loads require proper handling of data cleaning and conforming tasks – as planned in your ETL map.

If you create a solution that can handle only historical load of data your final score for this part of the MP is highly limited, you can get a passing grade, but it is limited to 65%. Moreover, solutions which do not introduce two separate solutions (understood, as two completely different systems that are triggered separately) to handle historical and incremental loads are preferred.

## **Task 2.1 – ETL SYSTEM – Solutions**:

ETL implementation – using a tool of your choice (suggested SSIS and MS SQL Server, remember about staging the data):

1. Divide your data into three batches (first batch can be, and should be, significantly larger)
2. Historical load of data – with the first batch
   1. Define target data structures – required dimension tables and a fact table
   2. Perform extraction of needed data from the source system. Please consider limiting the extraction to small portions of data (like 10000 rows), you’ll that this is a required approach for larger datasets (required for **>85%**).
   3. Perform basic anomaly detection and resolution transformations. Define a basic set of data-quality screens and implement them (and further run against extracted data). Please note that a simple data-quality screen can check for null values and mark the rows as requiring attention. If you are using SSIS, please note that more advanced screens can be implemented using external scripts. In short, define an approach and clean the identified anomalies
   4. Perform needed data transformation - creation of assumed multi-dimensional scheme - star, snowflake, or fact constellation:
      1. Create dimension tables. Remember to integrate and standardize the needed information, to make names and values verbose, to introduce surrogate keys, to handle SCDs, please also include basic sanity checks against the completeness and correctness of data, e.g., row counts. Consider implementing automated tests (required for **>85%**).
      2. Create fact table. Remember to integrate and standardize the needed information. Please also include basic sanity checks against the completeness and correctness of data, e.g., row counts. Consider implementing automated tests (required for **>85%**).
   5. Finally load the prepared data into target tables.
      1. Define proper relations between dimensions and fact tables.
   6. Use proper logging mechanisms and proper error handling (required for **>85%**). Remember to capture all erroneous data.
3. Incremental load of data – with the second and third batch
   1. Identify required extraction approach and implement it – remember to utilise a mechanism to capture only new/modified data (required for **>65%**). Perform extraction of needed data from the source system
   2. Perform basic anomaly detection and resolution transformations. Define a basic set of data-quality screens and implement them (and further run against extracted data). Please note that a simple data-quality screen can check for null values and mark the rows as requiring attention. If you are using SSIS, please note that more advanced screens can be implemented using external scripts. Consider implementing automated quality tests (required for >**50%)**. In short, define an approach and clean the identified anomalies.
   3. Perform needed data transformation - creation of assumed multi-dimensional scheme - star, snowflake, or fact constellation. Create dimension and fact tables. Remember Implement a mechanism to update only new/modified data (required for **>65%**)
   4. Finally load the prepared data into target tables.
   5. Use proper logging mechanisms and proper error handling (required for **>85%**). Remember to capture all erroneous data.

As the final solution, please upload your source files. In the final report provide a concise description of your implementation in accordance with the following points:

### Target database creation script

Please specify target database creation script:

-- DimDate table

CREATE TABLE DimDate (

Date DATE PRIMARY KEY,

Year INT,

Month INT,

MonthName VARCHAR(20),

Day INT

);

-- DimCircuit table

CREATE TABLE DimCircuit (

CircuitID INT PRIMARY KEY,

CircuitRef VARCHAR(50),

Name VARCHAR(100),

City VARCHAR(50),

Country VARCHAR(50)

);

-- DimConstructor table

CREATE TABLE DimConstructor (

ConstructorID INT PRIMARY KEY,

ConstructorRef VARCHAR(50),

Name VARCHAR(100),

Nationality VARCHAR(50)

);

-- DimDriver table

CREATE TABLE DimDriver (

DriverID INT PRIMARY KEY,

DriverRef VARCHAR(50),

Number VARCHAR(10),

Code VARCHAR(10),

Forename VARCHAR(50),

Surname VARCHAR(50),

DOB DATE,

Nationality VARCHAR(50)

);

-- DimRace

CREATE TABLE DimRace (

RaceID INT PRIMARY KEY,

Raceound INT,

RaceDate DATE,

);

-- DimWeather

CREATE TABLE DimWeather (

WeatherID INT PRIMARY KEY,

TempMax FLOAT, -- °C

TempMin FLOAT, -- °C

TempAvg FLOAT, -- °C

FeelsLikeMax FLOAT, -- °C

FeelsLikeMin FLOAT, -- °C

FeelsLikeAvg FLOAT, -- °C

DewPoint FLOAT, -- °C

Humidity FLOAT, -- %

Precipitation FLOAT, -- mm

PrecipitationCover FLOAT,-- %

SnowDepth FLOAT, -- cm

WindGust FLOAT, -- km/h

WindSpeed FLOAT, -- km/h

WindDirection INT, -- °

SeaLevelPressure FLOAT, -- hPa

Visibility FLOAT, -- km

Description VARCHAR(255) -- text

);

### ETL Processes

Please specify stages of your implemented approach and provided detailed description of each stage, provide details for each target table. Please specify tools that you have used and highlight their purpose in the process.

**Extract**

F1 Ergast API: Excel files downloaded from zip folder in api documentation with latest data

WEATHER API Python script: weather-data-fetch.py extracts weather data from visual crossing API

import time

import requests

import pandas as pd

from io import StringIO

# Read the API key

api\_key = 'SEPZQXTCDC9BPRVCNAGVATCS9'

# File path to the CSV file

file\_path = r'C:\Users\ishii\Documents\Formula-1-And-Weather-Data-Engineering-Project\Weather-Visual-Crossing-API-Data\New-Data.csv'

# Load the CSV file

try:

    df = pd.read\_csv(file\_path)

    print("Column names in the CSV file:", df.columns.tolist())

except FileNotFoundError:

    print(f"Error: The file at {file\_path} was not found.")

    exit(1)

except pd.errors.EmptyDataError:

    print(f"Error: The file at {file\_path} is empty.")

    exit(1)

# Define the weather columns to keep

weather\_columns = [

    'tempmax', 'tempmin', 'temp', 'feelslikemax', 'feelslikemin', 'feelslike',

    'dew', 'humidity', 'precip', 'precipprob', 'precipcover', 'preciptype',

    'snow', 'snowdepth', 'windgust', 'windspeed', 'winddir', 'sealevelpressure',

    'cloudcover', 'visibility', 'solarradiation', 'solarenergy', 'uvindex',

    'severerisk', 'conditions', 'description', 'icon', 'stations'

]

# Add the weather columns to the DataFrame with default NaN values

for column in weather\_columns:

    if column not in df.columns:

        df[column] = pd.NA

# Function to fetch weather data for a given location and date with retries

def fetch\_weather\_data(location, date):

    formatted\_date = date.strftime('%Y-%m-%d')

    url = f'https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/{location}/{formatted\_date}/{formatted\_date}'

    params = {

        'unitGroup': 'metric',

        'contentType': 'csv',

        'include': 'days',

        'key': api\_key

    }

    retries = 5

    for i in range(retries):

        response = requests.get(url, params=params)

        if response.status\_code == 200:

            return pd.read\_csv(StringIO(response.text))

        elif response.status\_code == 429:

            # Exponential backoff with a maximum of 32 seconds

            sleep\_time = min(2 \*\* i, 32)

            print(f"Rate limit hit. Retrying in {sleep\_time} seconds...")

            time.sleep(sleep\_time)

        else:

            print(f"Error fetching data for {location} on {formatted\_date}: {response.status\_code}")

            return pd.DataFrame(columns=weather\_columns)

    return pd.DataFrame(columns=weather\_columns)

# Iterate over each row in the DataFrame to fetch and update weather data

for index, row in df.iterrows():

    location = row['locationModified'].replace('%20', ' ')

    date = pd.to\_datetime(row['date']).date()

    weather\_df = fetch\_weather\_data(location, date)

    if not weather\_df.empty and weather\_df.notna().any().any():

        for column in weather\_columns:

            if column in weather\_df.columns:

                df.at[index, column] = weather\_df[column].iloc[0]

    time.sleep(1)  # Constant delay to avoid hitting API rate limits

# Save the updated DataFrame back to the same CSV file

df.to\_csv(file\_path, index=False)

print(f"Weather data fetched and updated to {file\_path}")

**Batching process:**

*We will extract the batches using year from date where:*

*- batch 1 = 1950 – 2000,*

*- batch 2 = 2001 – 2011,*

*- batch 3: 2012 – 2024*

*Both races and weather contain date columns so that is easy to process:*

*- date (column) -> races (csv) (batch 1: 1950 – 2000, batch 2: 2001 – 2011, batch 3: 2012 – 2024)*

*- date (column) -> weather-data (csv) (batch 1: 1950 – 2000, batch 2: 2001 – 2011, batch 3: 2012 – 2024)*

*A screenshot of a computer

Description automatically generated*

*For the rest of the dataset we will use the relational database relationship to connect and create the batches: (races, circuits, results, pit\_stops, lap\_times, driver\_standings, constructor\_standings, constructor, drivers csv file) (circuitId, raceId, constructorId, driverId - columns)*

*- races(circuitId) -> circuits(circuitId)*

*- races(raceId) -> results(raceId)*

*- races(raceId) -> pit\_stops(raceId)*

*- races(raceId) -> lap\_times(raceId)*

*- races(raceId) -> driver\_standings(raceId)*

*- races(raceId) -> constructor\_standings (raceId)*

*- results(constructorId) -> constructor(constructorId)*

*- results(driverId) -> drivers(driverId)*

*- results(statusId) -> status(statusId)*

*Tools: Python script - batch-processing.py*

Staging database: Creating relational tables in SSIS

A computer screen shot of a diagram

Description automatically generated

CREATE TABLE Circuit (

CircuitID INT PRIMARY KEY,

CircuitRef VARCHAR(50),

Name VARCHAR(100),

city VARCHAR(50),

Country VARCHAR(50)

);

CREATE TABLE Race (

RaceID INT PRIMARY KEY,

CircuitID INT,

Raceround INT,

RaceDate DATE,

CONSTRAINT FK\_RaceICircuit FOREIGN KEY (CircuitId) REFERENCES Circuit (CircuitId)

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE TABLE Constructor (

ConstructorID INT PRIMARY KEY,

ConstructorRef VARCHAR(50),

Name VARCHAR(100),

Nationality VARCHAR(50)

);

CREATE TABLE Driver (

DriverID INT PRIMARY KEY,

DriverRef VARCHAR(50),

Number VARCHAR(10),

Code VARCHAR(10),

Forename VARCHAR(50),

Surname VARCHAR(50),

Nationality VARCHAR(50)

);

CREATE TABLE Status(

statusId INT PRIMARY KEY,

satus VARCHAR(50)

)

CREATE TABLE Weather (

weatherId INT PRIMARY KEY,

circuitId INT,

date DATE,

TempMax FLOAT, -- °C

TempMin FLOAT, -- °C

TempAvg FLOAT, -- °C

FeelsLikeMax FLOAT, -- °C

FeelsLikeMin FLOAT, -- °C

FeelsLikeAvg FLOAT, -- °C

DewPoint FLOAT, -- °C

Humidity FLOAT, -- %

Precipitation FLOAT, -- mm

PrecipitationCover FLOAT,-- %

SnowDepth FLOAT, -- cm

WindGust FLOAT, -- km/h

WindSpeed FLOAT, -- km/h

WindDirection INT, -- °

SeaLevelPressure FLOAT, -- hPa

Visibility FLOAT, -- km

Description VARCHAR(255) -- text

CONSTRAINT FK\_WeatherCircuit FOREIGN KEY (CircuitId)

REFERENCES Circuit (CircuitId)

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE TABLE DriverStanding(

driverStandingIdId INT PRIMARY KEY,

raceId INT,

driverID INT,

position INT,

CONSTRAINT FK\_DriverStandingDriver FOREIGN KEY (driverId)

REFERENCES Driver (driverId)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_DriverStandingRace FOREIGN KEY (RaceId)

REFERENCES Race (RaceId)

ON DELETE CASCADE

ON UPDATE CASCADE

)

CREATE TABLE ConstructorStanding(

constructorStandingId INT PRIMARY KEY,

raceId INT,

constructorId INT,

position INT,

CONSTRAINT FK\_ConstructorStandingDriver FOREIGN KEY (constructorId)

REFERENCES Constructor (constructorId)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_ConstructorStandingRace FOREIGN KEY (raceId)

REFERENCES Race (raceId)

ON DELETE CASCADE

ON UPDATE CASCADE

)

CREATE TABLE Result(

resultId INT,

raceId INT,

driverId INT,

constructorId INT,

fastestLap INT,

fastestLapSpeed DECIMAL,

statusId INT,

CONSTRAINT FK\_ResulConstructor FOREIGN KEY (constructorId)

REFERENCES Constructor (constructorId)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_ResultRace FOREIGN KEY (raceId)

REFERENCES Race (raceId)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_ResultDriver FOREIGN KEY (driverId)

REFERENCES Driver (driverId)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_ResultStatus FOREIGN KEY (statusId)

REFERENCES Status (statusId)

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE TABLE Pit\_stops (

raceId INT,

driverId INT,

stop INT,

lap INT,

time TIME,

duration TIME,

milliseconds INT,

PRIMARY KEY (raceId, driverId, stop),

FOREIGN KEY (raceId) REFERENCES Race(RaceID)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (driverId) REFERENCES Driver(DriverID)

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE TABLE Lap\_times (

raceId INT,

driverId INT,

lap INT,

position INT,

time TIME,

milliseconds INT,

PRIMARY KEY (raceId, driverId, lap),

FOREIGN KEY (raceId) REFERENCES Race(RaceID)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (driverId) REFERENCES Driver(DriverID)

ON DELETE CASCADE

ON UPDATE CASCADE

);

Loading flat files into staging relational database

Transform

*1. There were inconsistencies found in the tables with ‘date’ attribute – example formats fount: ‘3/7/1999 12:00:00 AM’, ‘2/2/2023’, ‘2002-03-03’. Date in each CSV is transformed into correct date format i.e. ‘dd-mm-yyyy’.*

Load

### General approach

Please specify stages of your approach and provided detailed description of each stage.

**…**

### Dimension [X]

For each dimension [X], please specify stages of your approach and provided detailed description of each stage.

**…**

### FACT [X]

For each fact [X], please specify stages of your approach and provided detailed description of each stage.

**…**

## **Task 2.2 – CUBE model – details:**

Please prepare the model of the ETL process, in accordance with the below specification and discuss the solution with the lecturer.

*General note:* ***The complexity and completeness of the created cube affects the grade. During ongoing lab assignments and during lecture we have tackled different properties and settings related to proper OLAP cube definition and creation – try to use them and try to use them wisely.***

## **Task 2.2 – CUBE in SSAS – Solutions**:

1. Prepare a cube based on prepared data from the previous task.
   1. Remember to define needed dimensions, attributes, attribute relations, hierarchies, etc. Remember about proper ordering of members within the dimensions – e.g., January comes before February
   2. Define needed measures and aggregation functions, try including at least one calculated measure
   3. Additionally define (*required for 85%*): at least a single perspective, at least one KPI and proper aggregations (cube materialisations)
2. Process and deploy your cube in your local SSAS instance
   1. Be able to show and document your results in cube’s browser

As the final solution, please upload your source files. In the final report provide a concise description of your implementation in accordance with the following points:

### 2.2.1 Final cube structure

Prepare a screenshot with your final cube structure (from SSAS).

…

### 2.2.2 Measures

Prepare a short description and a screenshot for all measures (include information about formatting string and aggregation function).

…

### 2.2.3 Dimensions

Prepare a short description and a screenshot for all dimension structures. Remember to include information about hierarchies, attribute relations, and additional processing within the cube (e.g., grouping, ordering, key definitions).

…

### 2.2.4 Cube details

Prepare a short description and a screenshot for all additional mechanisms utilised, e.g., calculations, KPIs, aggregations, partitions, perspectives.

…

### 2.2.5 Process results

Prepare a screenshot from successful completion of cube processing and deployment process, and a screenshot from cube browser (with an exemplar query of your choice – make sure to capture cube’s structure).

…

## **General Conclusions**:

Use this section to provide your general conclusions:

…

Remarks:

* A report without final conclusions will not be checked and results in a negative score!
* You should use MS SQL SERVER 20XX, SQL Server Integration Services (SSIS) and SQL Server Analysis Services (SSAS); if needed you can use additional scripts or software to develop the ETL solution (usage of dedicated libraries or tools is encouraged)

Examples from *J. Mundy, W. Thornthwaite, R. Kimball, The Microsoft Data Warehouse Toolkit: With SQL Server 2005 and the Microsoft Business Intelligence Toolset, Wiley, 2006*

*Dimensions:*

Diagram

Description automatically generated

*Facts:*

Diagram

Description automatically generated