Sr Data & Reporting Engineer Tasks/Questions

We expect this to take no more than 8 hours of work. If you believe this estimation is off please

provide us with your estimation along with an explanation.

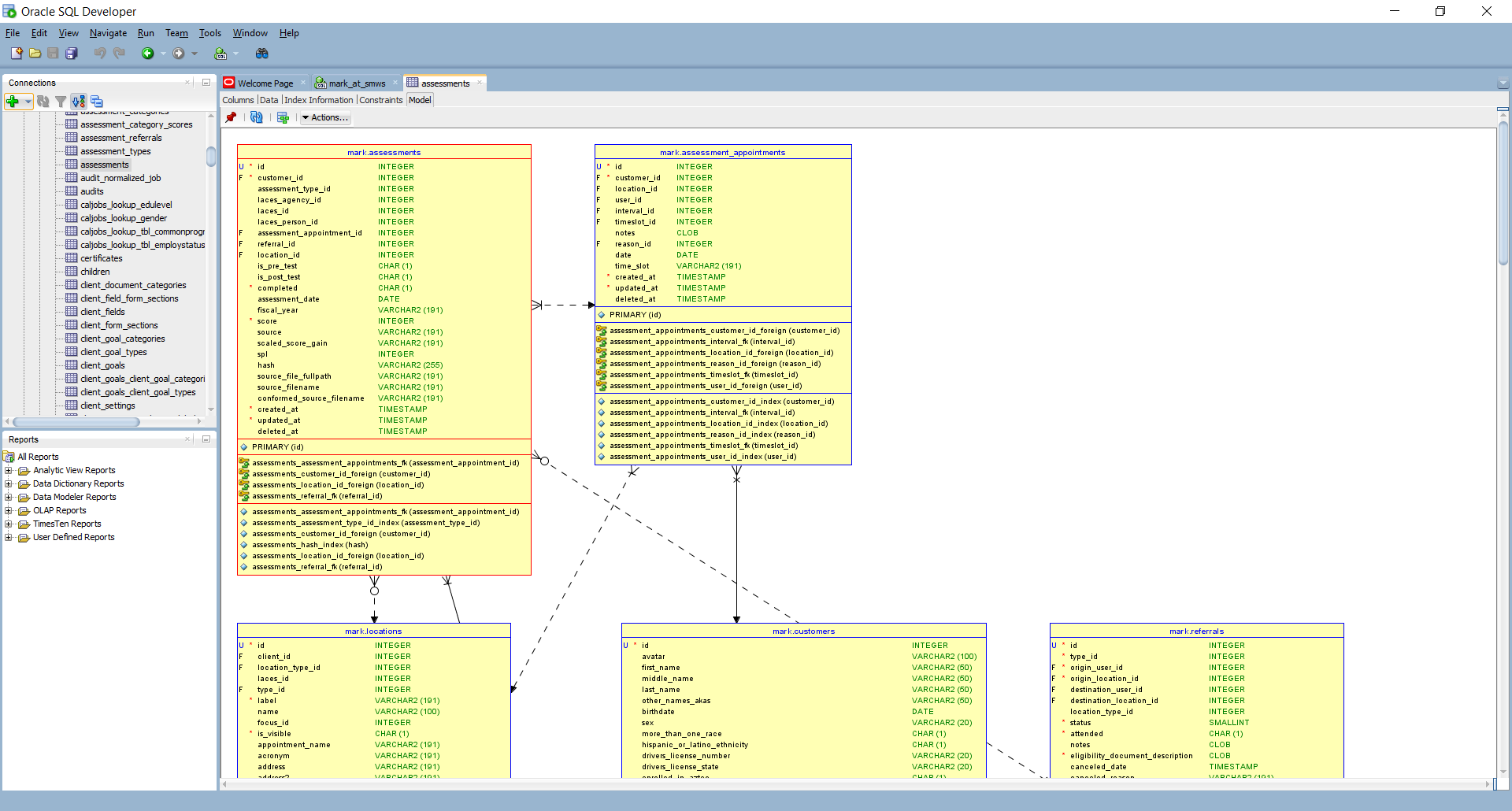
Analytics and reporting structure and visualization

● Take the given DB backup and import it to a MySQL database.

Imported to local MySQL database. (My AWS demo account expired long ago)

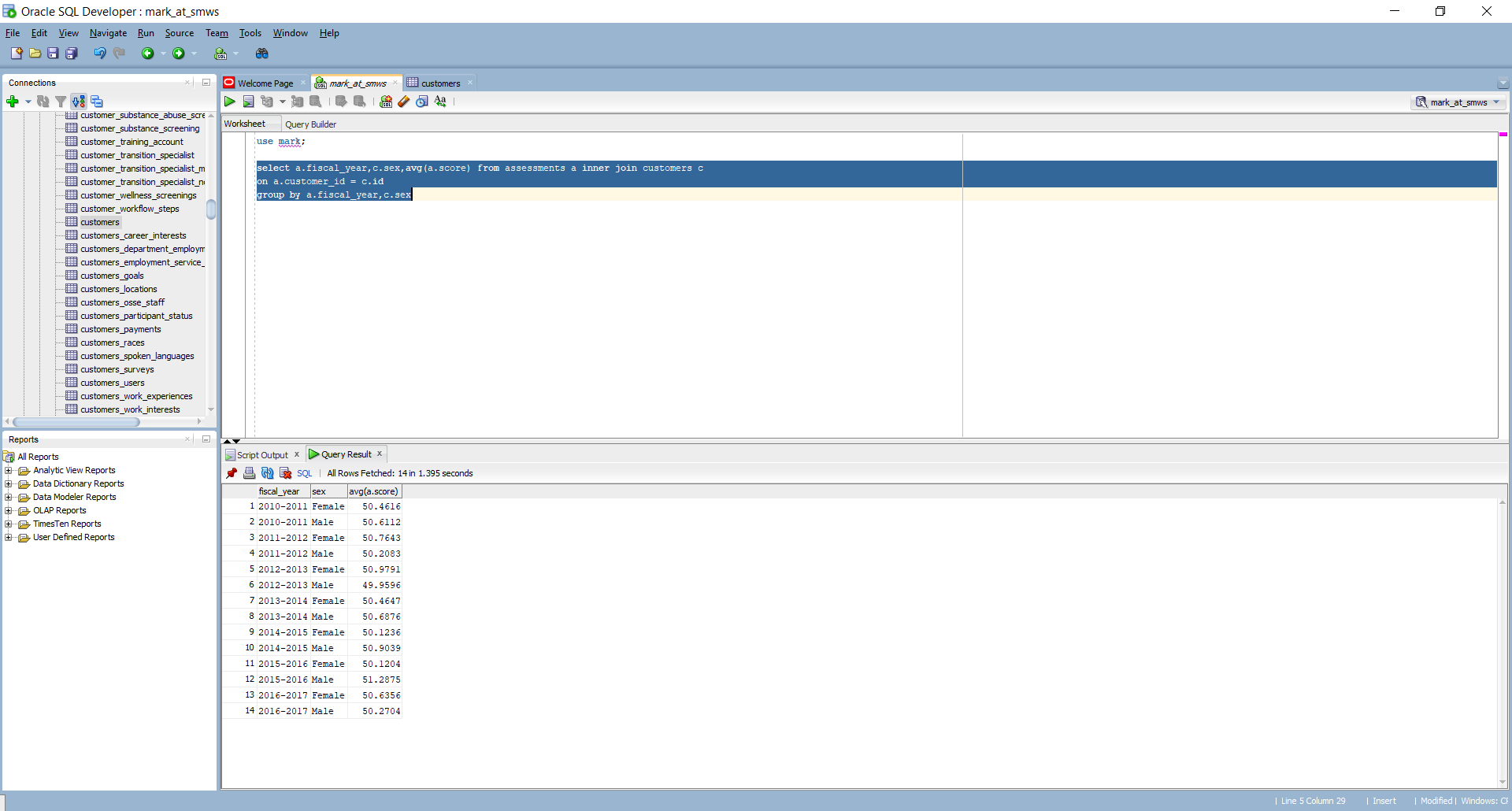
● Create an analytical database structure using a database of your choice.

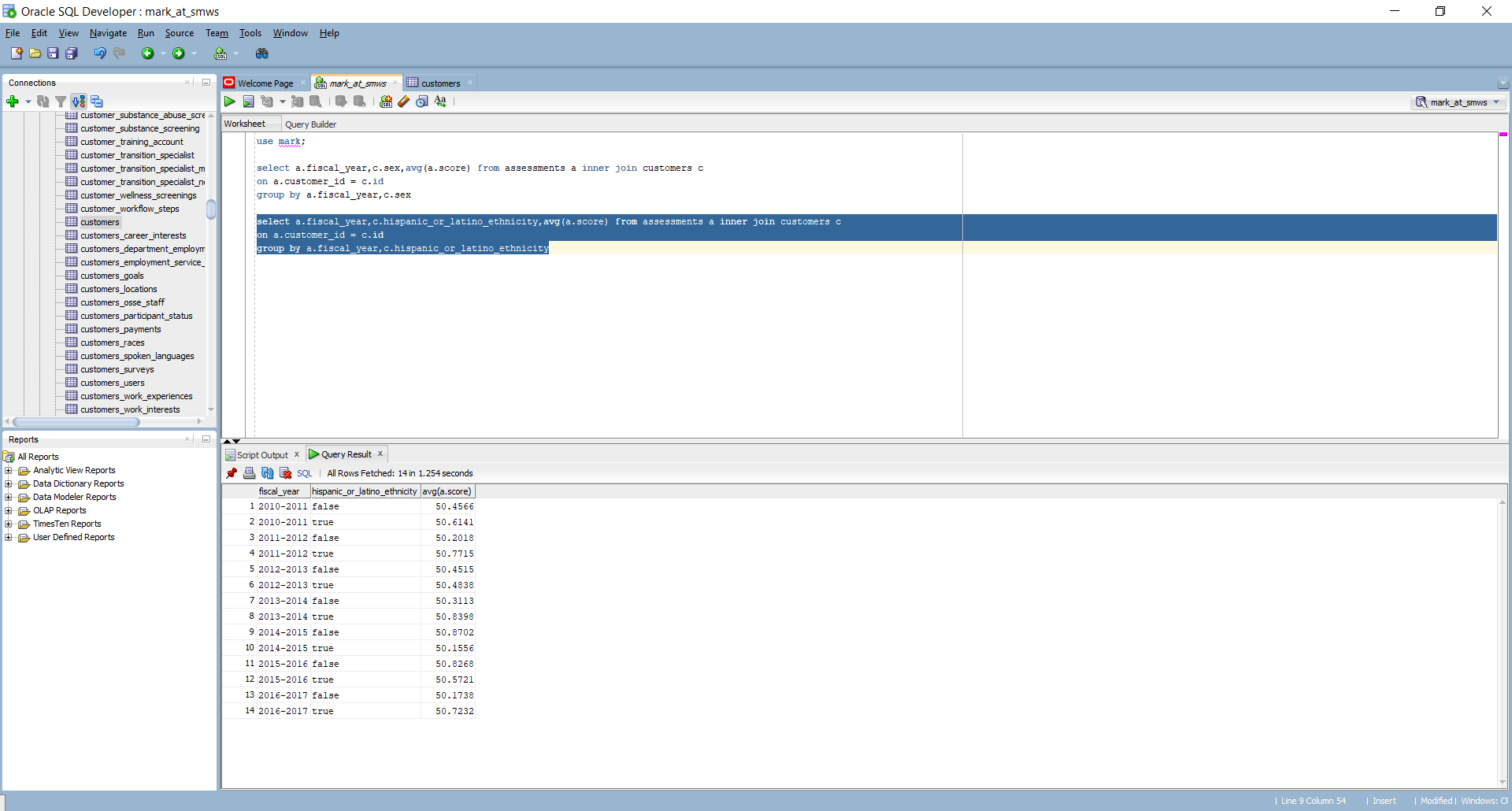
Started by reverse-engineering a physical model from the database centered on the ASSESSMENTS table:



Preparing for the question below, we see that the independent variables for analysis (sex, hispanic\_or\_latino\_ethnicity, more\_than\_one\_race) reside in the CUSTOMERS table.

We perform a few queries against the source data to check values and model sanity:

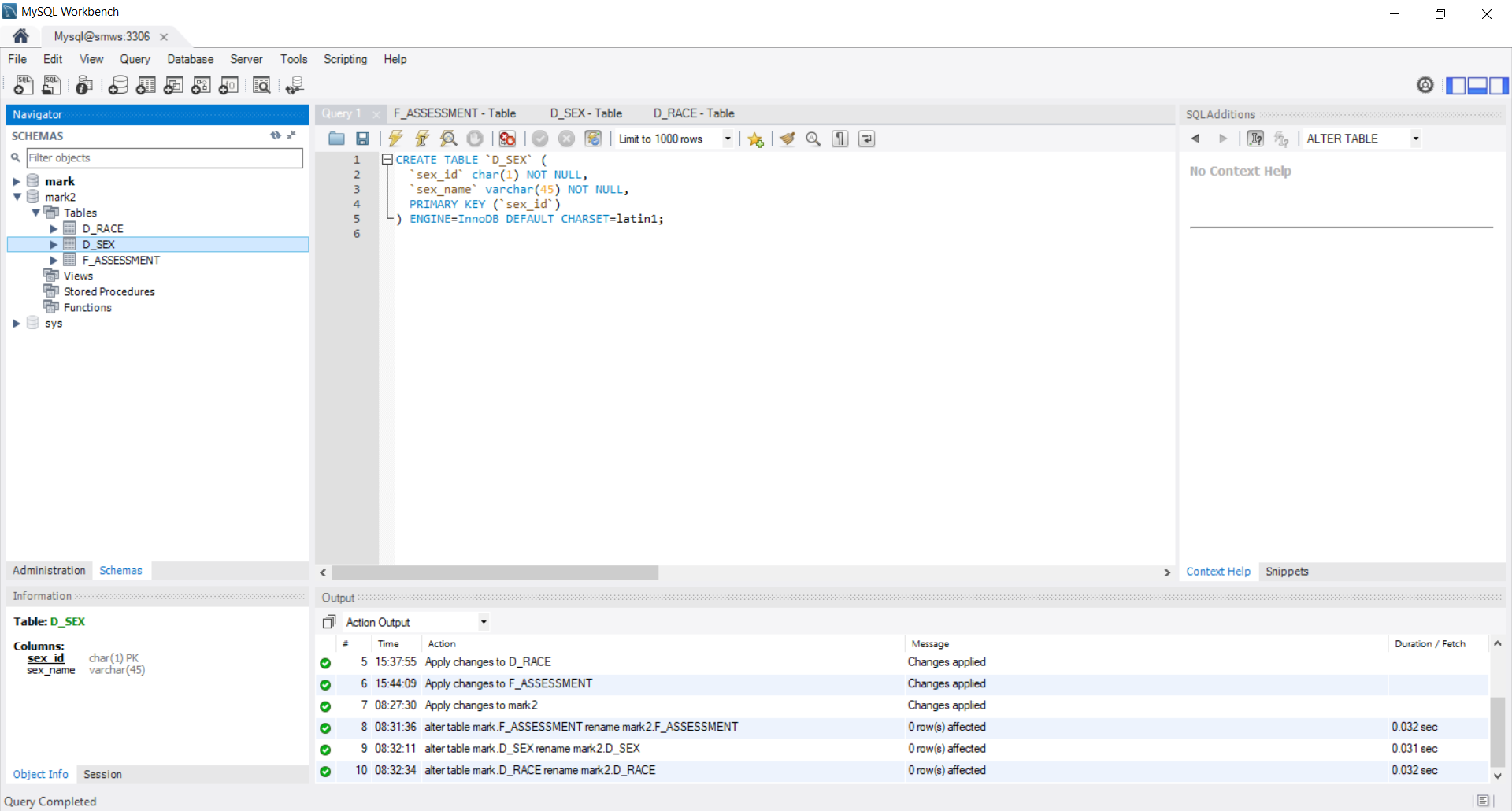


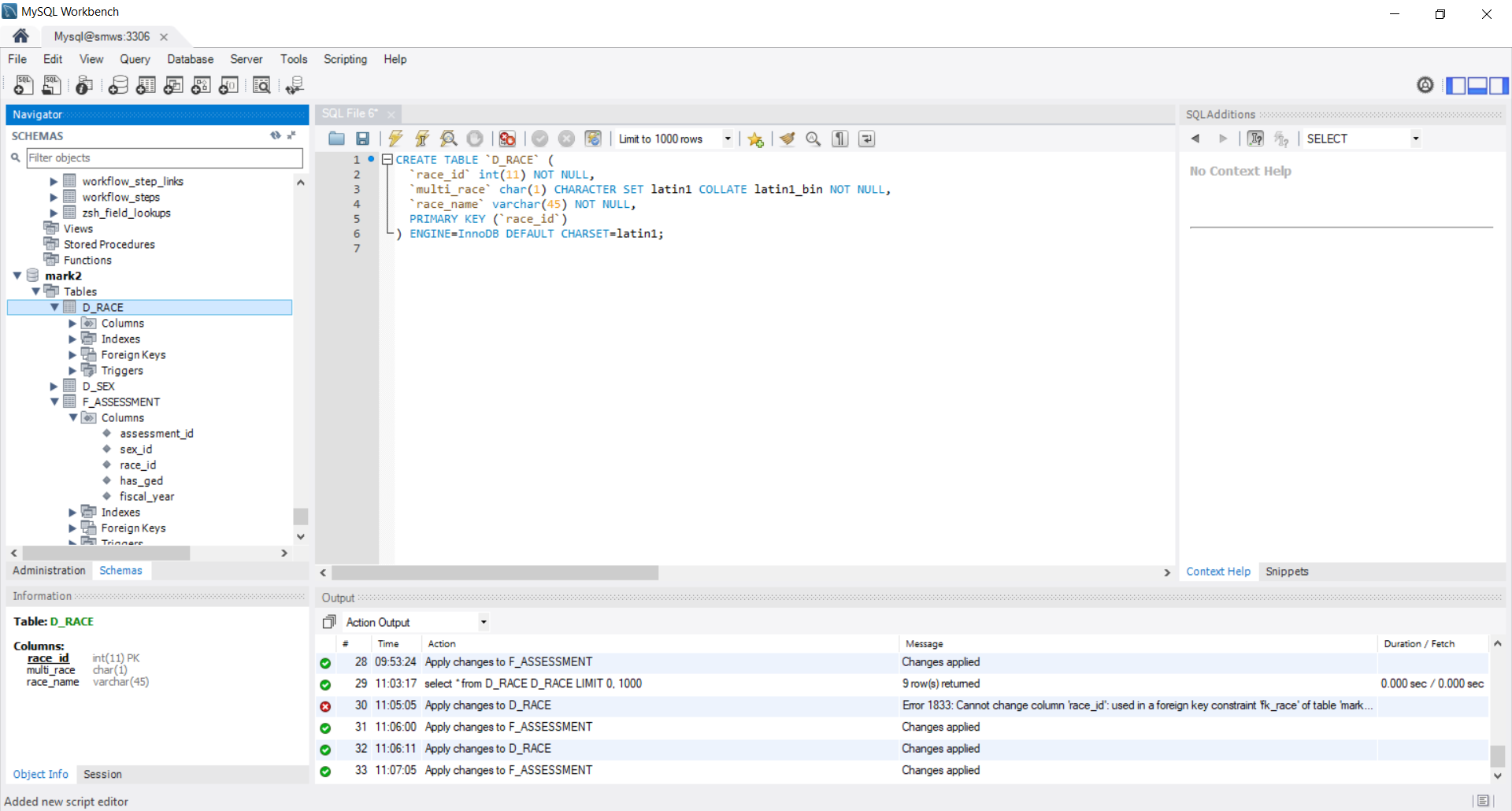


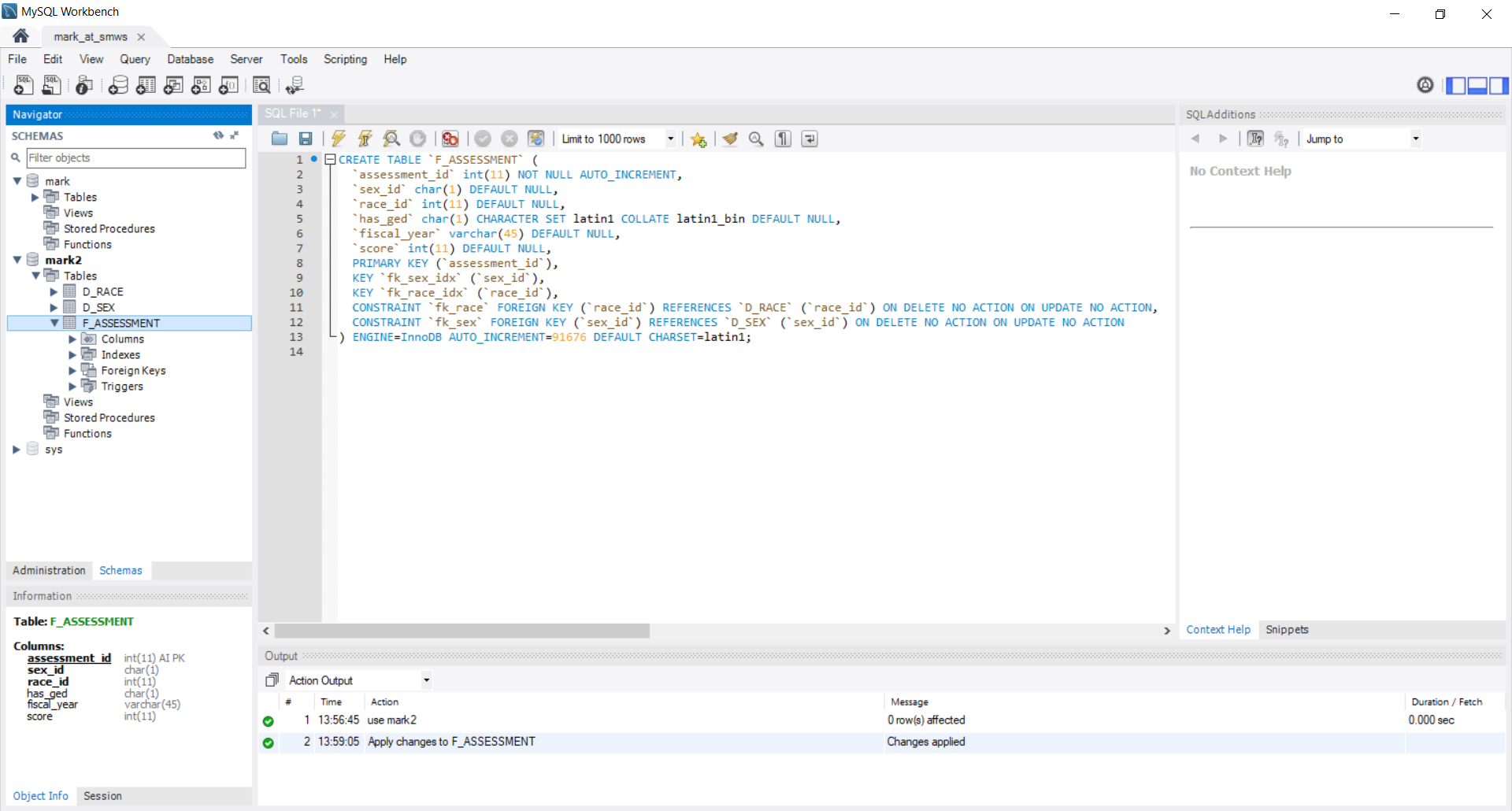
● Using tooling of your choice, transfer necessary data from the application/transactional

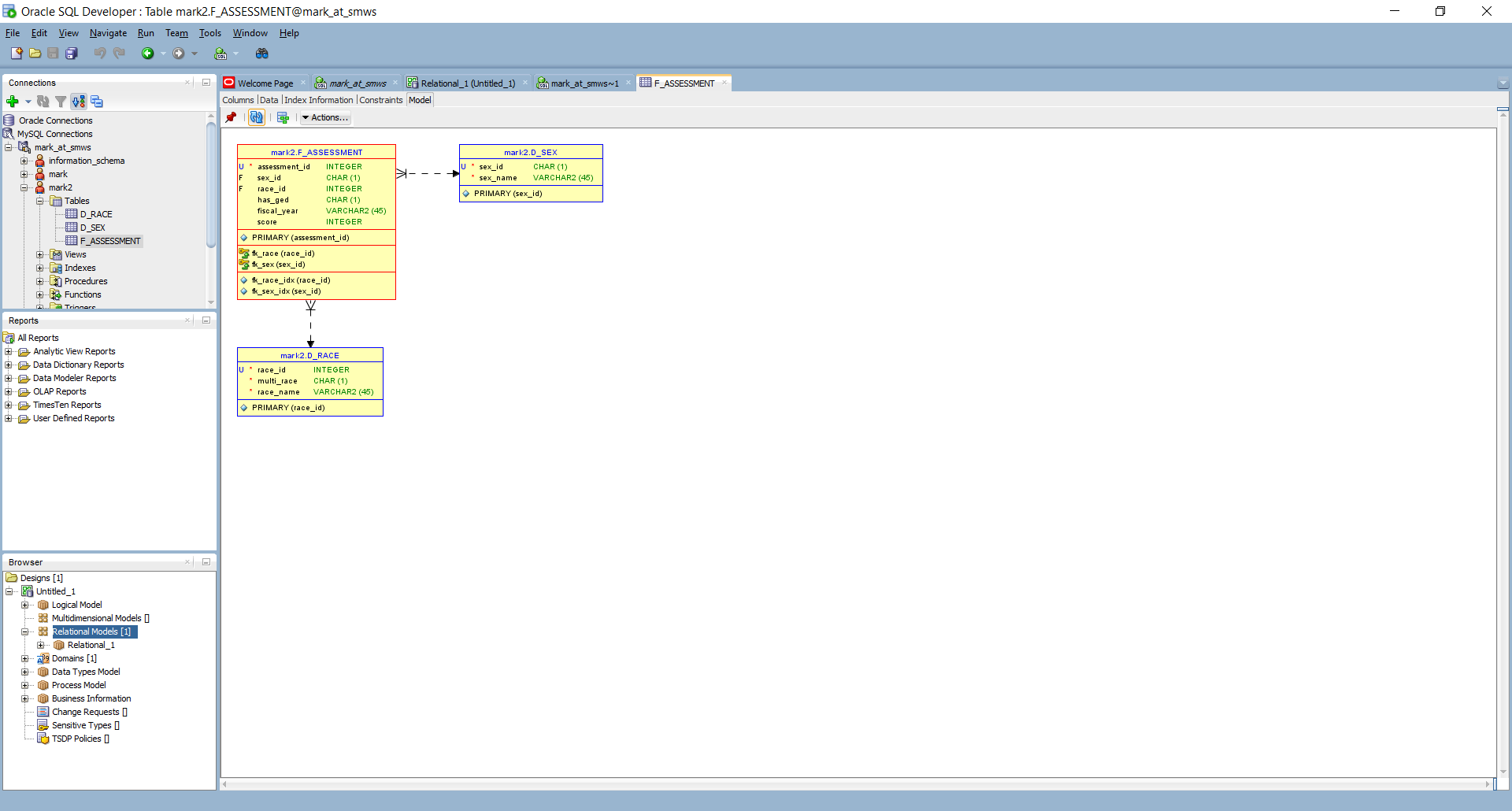
database to the analytical database.

Let's create a small type-1 dimensional model to demonstrate ETL.







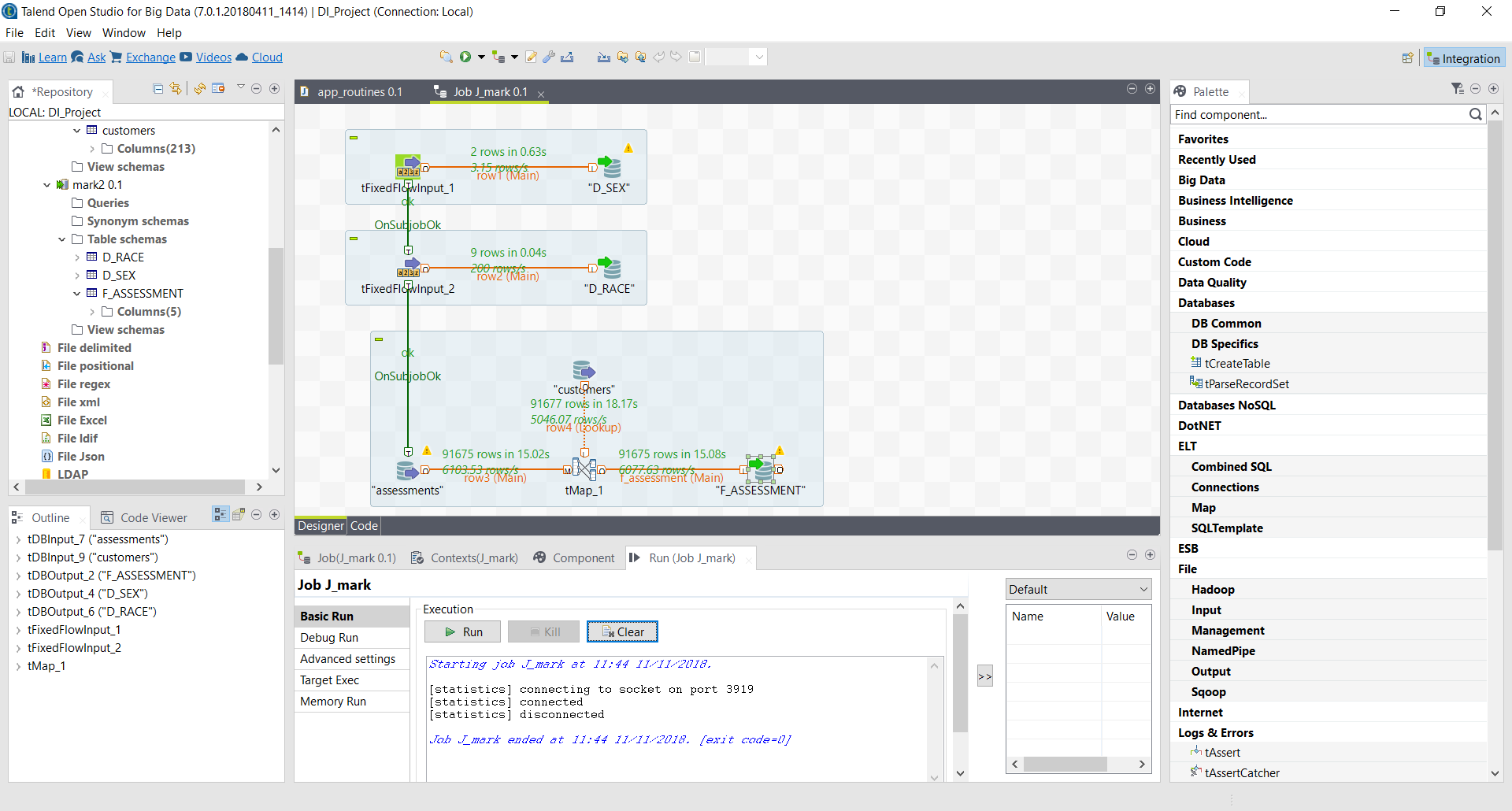


create view assessment\_by\_ged as select `mark2`.`F\_ASSESSMENT`.`fiscal\_year` AS `fiscal\_year`,if((`mark2`.`F\_ASSESSMENT`.`has\_ged` = 1),'Y','N') AS `ged`,avg(`mark2`.`F\_ASSESSMENT`.`score`) AS `avg(score)` from `mark2`.`F\_ASSESSMENT` group by `mark2`.`F\_ASSESSMENT`.`fiscal\_year`,`mark2`.`F\_ASSESSMENT`.`has\_ged`

create view assessment\_by\_race as select `mark2`.`F\_ASSESSMENT`.`fiscal\_year` AS `fiscal\_year`,`mark2`.`D\_RACE`.`race\_name` AS `race\_name`,avg(`mark2`.`F\_ASSESSMENT`.`score`) AS `avg(score)` from (`mark2`.`F\_ASSESSMENT` join `mark2`.`D\_RACE` on((`mark2`.`F\_ASSESSMENT`.`race\_id` = `mark2`.`D\_RACE`.`race\_id`))) group by `mark2`.`F\_ASSESSMENT`.`fiscal\_year`,`mark2`.`D\_RACE`.`race\_name`

create view assessment\_by\_sex as select `mark2`.`F\_ASSESSMENT`.`fiscal\_year` AS `fiscal\_year`,`mark2`.`F\_ASSESSMENT`.`sex\_id` AS `sex\_id`,avg(`mark2`.`F\_ASSESSMENT`.`score`) AS `avg(score)` from `mark2`.`F\_ASSESSMENT` group by `mark2`.`F\_ASSESSMENT`.`fiscal\_year`,`mark2`.`F\_ASSESSMENT`.`sex\_id`

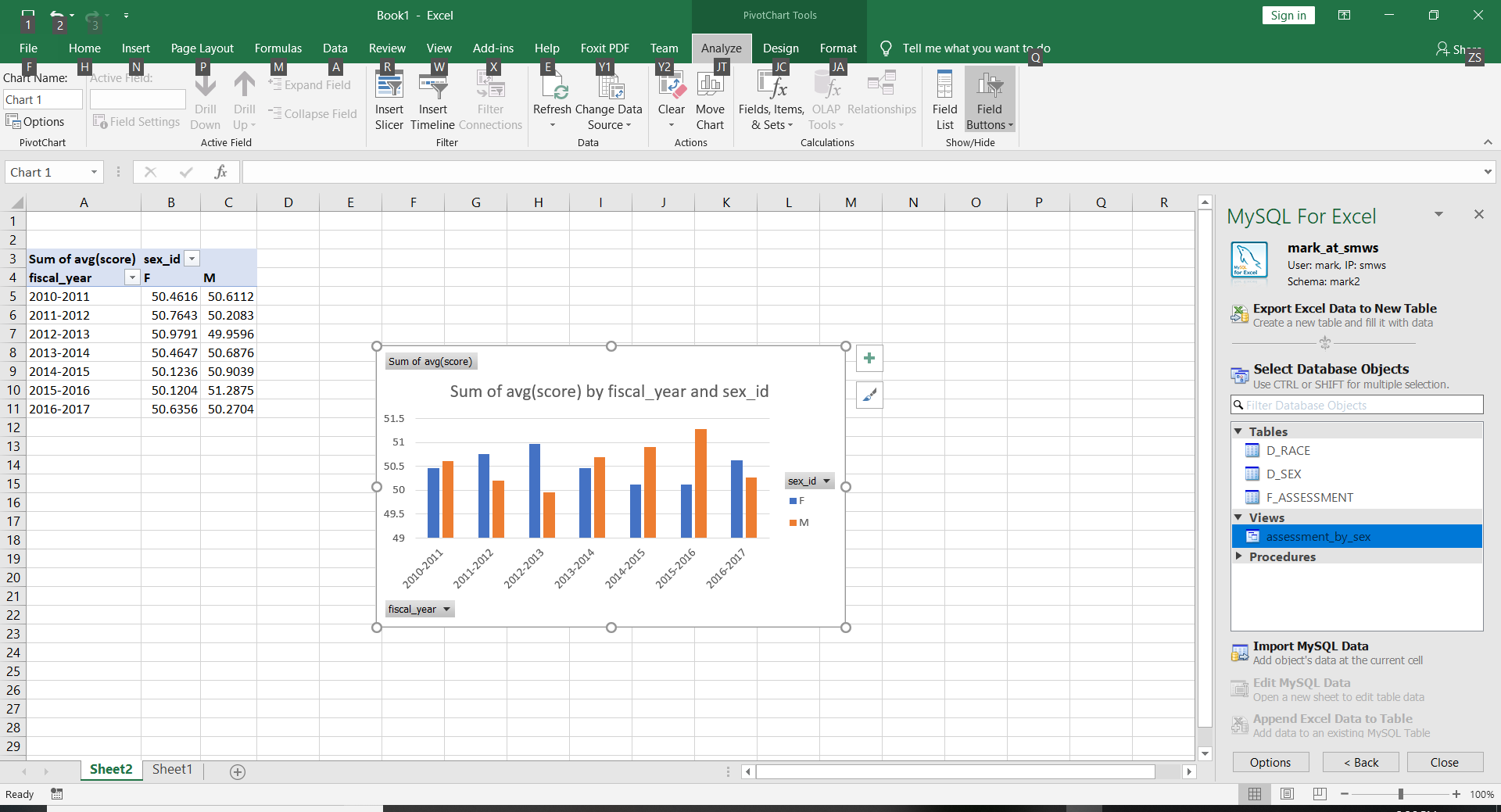
Load the analytical database. This is a simplified approach that does not accommodate Slowly Changing Dimensions. I've placed an export of this Talend project in the GitHub repo, but you will need a Talend Open Studio installation (A free download) to browse it.

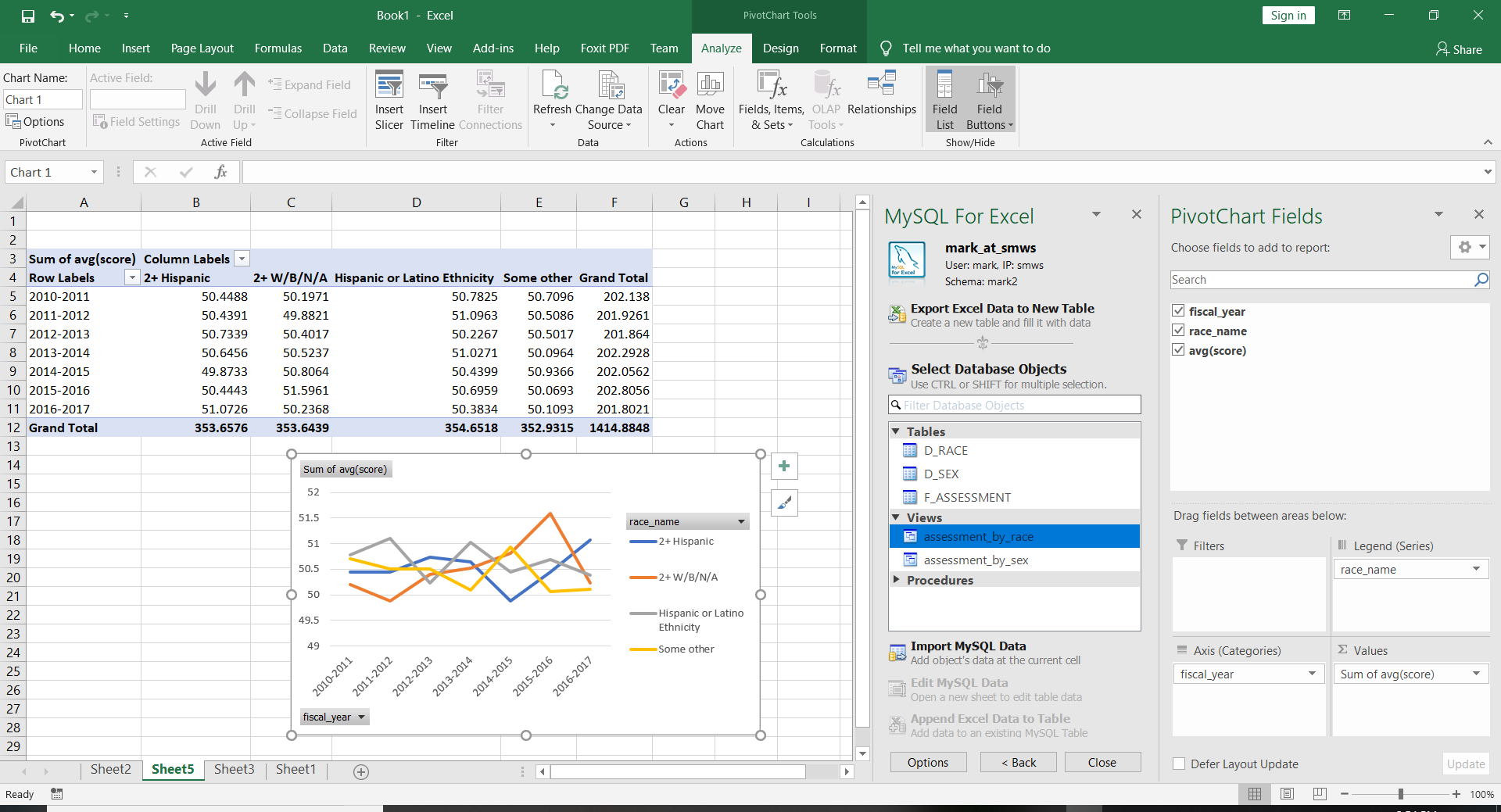


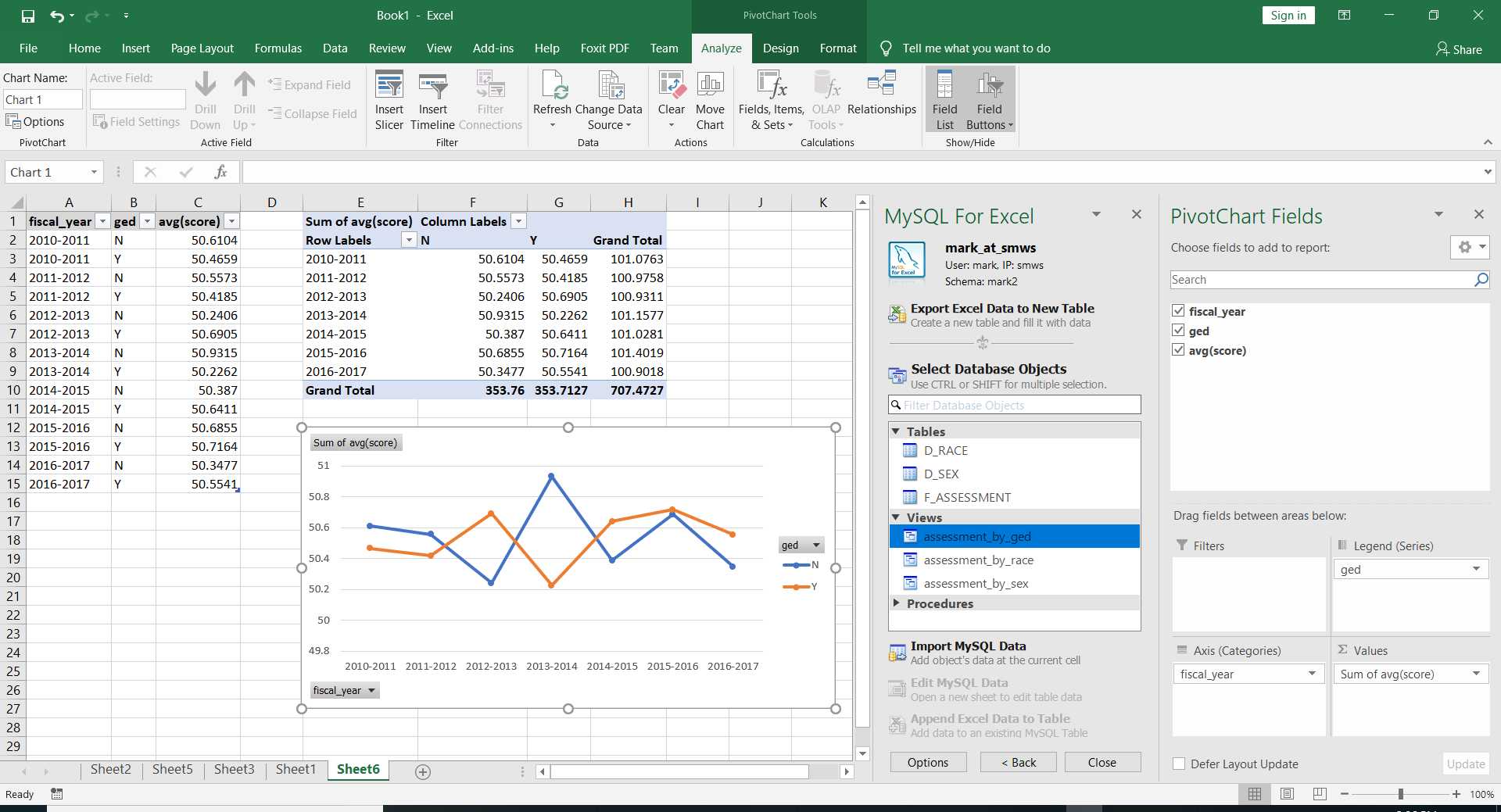
● Using tooling of your choice, create a report/visualization that shows/explains information

found about assessment scores and how they relate to gender, race, and whether the

customer has their GED(customers.has\_ged).







● Return to us:

○ A link to any code written for this task. We expect this to be hosted in a public

repo on GitHub.

○ A database backup of the analytical database after data has transferred to it.

○ A explanation of how we can run the code / use the tool selections

○ Screenshots and explanations are encouraged

DB query efficiency

select

'incoming' rec\_ind

,soft\_hash

,hard\_hash

,manual\_hash\_from\_cq

from

vie\_conform\_01\_hashed

where

manual\_hash\_from\_cq is not null

union all

select

'existing' rec\_ind

,customer\_soft\_hash

,customer\_hard\_hash

,manual\_hash

from

vie\_consolidate\_queue

where

manual\_hash in (select distinct manual\_hash from vie\_conform\_01\_hashed)

order by

2,3,1

Using the provided DB backup & the query above please:

● Provide metrics on this query as is.

The query executes on my home server in .016 seconds.

● Improve the query and explain your changes (We plan on testing the updated query on a

AWS RDS database instance with 2 vCPUs, 8 GB of memory, and a max of 1,000 IOPS

[db.m4.large]).

With such a small dataset, (10 rows for each table) query improvement suggestions would be speculative/academic until measured against greater volumes.

● Explain any considerations you took into account after reviewing the resource

constraints on the RDS instance.

I don't have access to an RDS instance.

● Provide metrics on your updated version of the query.

● Explain how you analyze poor performing queries on an AWS RDS instance.

I analyze poor performing queries in typical (row-oriented) relational databases by looking for indexing opportunities in relation to the cardinality of the columns and the type of queries being performed. For example, a high cardinality column used by queries with select criteria that qualify less than 1 to 10% of the rows can benefit by a b-tree index. Columns with low cardinality values are best used for bitmap indexing. When 5-10% of the rows or more are being selected, a full table scan is usually the best approach. Most good query optimizers will ignore indexes in this case. Query optimization strategies will vary according to the use case being either online, near real-time versus batch.

The tuning strategy is quite different for column-oriented and distributed storage architectures such as Redshift, Hadoop HDFS, and Snowflake. Older technologies such as HDFS and Redshift require careful attention to data distribution across the cluster.

Some databases require periodic housekeeping to maintain best performance. An example is the need for vacuuming with Redshift.

Most importantly, my query analysis is measurement-based.

DB Architecture

● Using the given DB backup, identify areas of the database that are poorly architected.

● Provide suggestions for how to fix those issues.

I'd need much more time and understanding of the data to comment on this large set of tables. However, for the visualization exercise above, I'd suggest establishing dimensions such as race to organize attributes such as "Hispanic" and "More\_than\_one\_race" found in the Customers table. I've included a simplified example in my materials.

3rd Party Source Systems

How can we work with 3rd party source systems to more easily get data from them? Describe

any experiences you've had with this and/or ideas that would facilitate the acquisition of data

from 3rd party source systems. Currently we are working with APIs and CSV files but the way

we get the files is varied based on source system.

Dealing with multiple formats and interfaces to 3rd party source systems is a fact of life unless you have significant "pull" with the source provider. For example, if you are a building products supplier, the specifications for data interfacing will be dictated to you by Home Depot. Some industries such as Healthcare provide some help with this via published standards. (eg HL7)

Where you don't have the "pull" to dictate, the best defense is to keep source-specific interfacing and data transformation as close to the endpoints as possible and keeping your most important and complex transformation and analysis "in the middle" and independent of source interface and document structure type.

Understanding Data

How would you integrate a master data management system to help track and understand

metadata for hundreds/thousands of data points from multiple source systems that we do not

control?

For example we collect a data point called "Highest education level upon entry" from various

systems but it carries a slightly different meaning depending on which system it comes from.

How can we best track our understanding of that data?

Master data management systems can assist with managing the meaning of attribute values and resolving conflict resolution for values from different source systems. But these master data management systems do not eliminate the need for human intervention when categorization and conflict resolution cannot be automated with rules-based logic. Human workflow is often a component of these systems. And sometimes, custom programming is required.

Therefore, I wouldn't presume a master data management system is the answer to this problem without a better understanding of the data, variation across sources, and the business rules for using it.