MONASH University

MONASH
INFORMATION
TECHNOLOGY

# FIT3003 – Business Intelligence and Data Warehousing

## Week 4 – Multi-Fact

Semester 2, 2022

Developed by:
Dr. Agnes Haryanto
Agnes.Haryanto@monash.edu

# Learning Objectives

1. To understand the concept of multi-fact star schema.
2. To be able to implement multi-fact star schema using SQL.
3. To understand the importance of data cleaning.
4. To be familiar with data exploration.
5. To be able to use SQL to do data exploration and data cleaning.

# Agenda

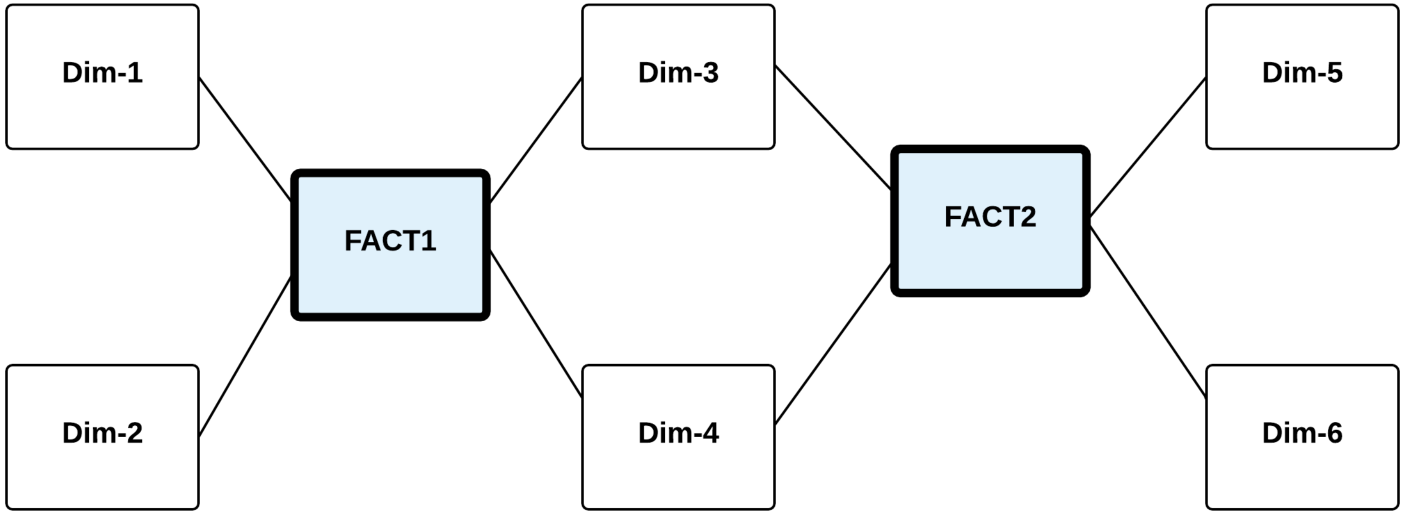
1. Multi-Fact
    1. Different Subject Multi-Fact
    2. Multi-Fact or Single Fact with multiple Fact Measures

2. Data Cleaning

# Multi-Fact

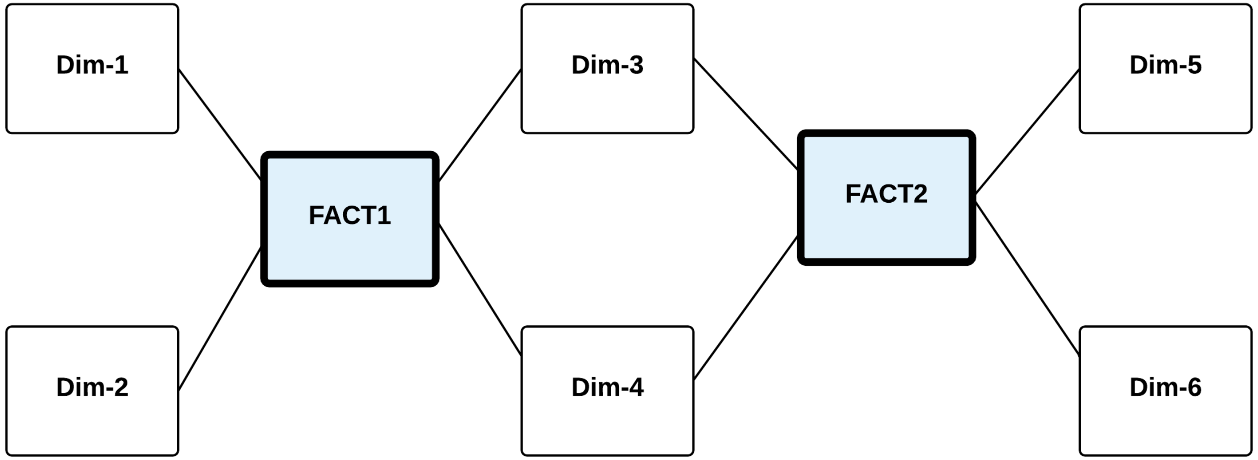
# Multi-Fact

- A *Subject-Oriented* data warehouse means that one star schema focuses on one subject only.
  - A subject refers to a topic of analysis.

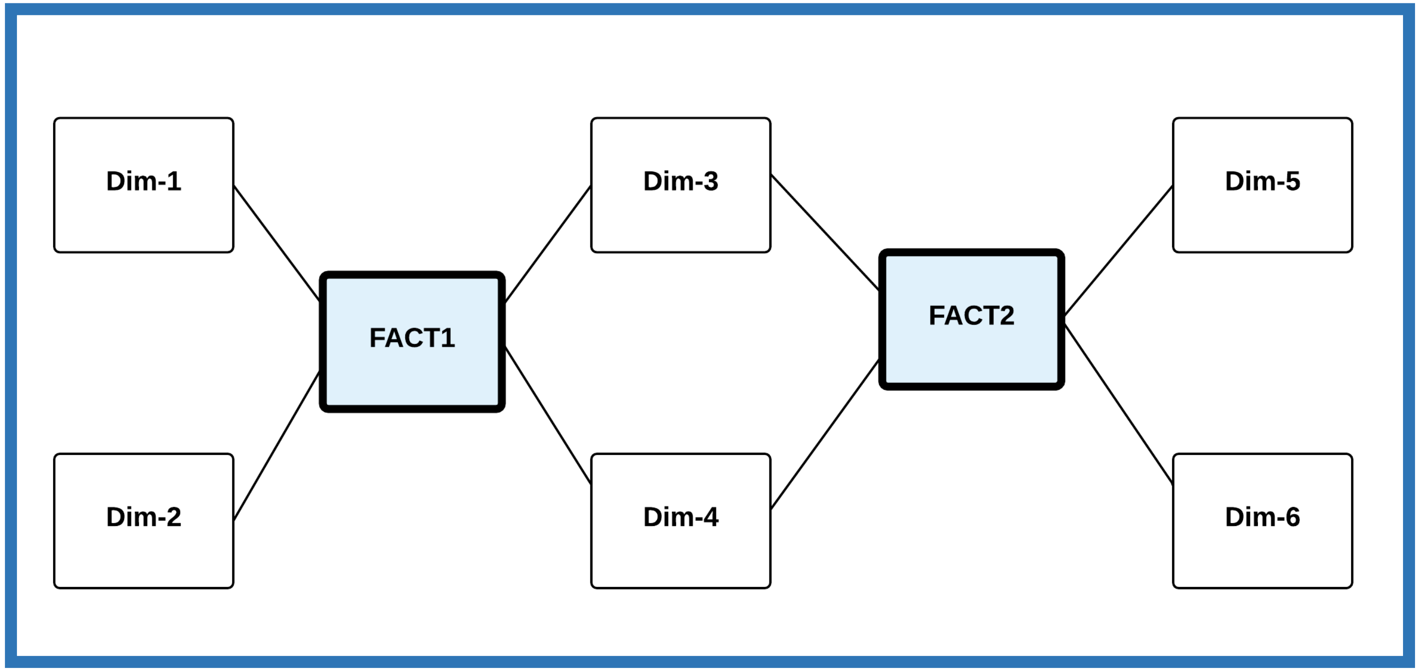


# Multi-Fact

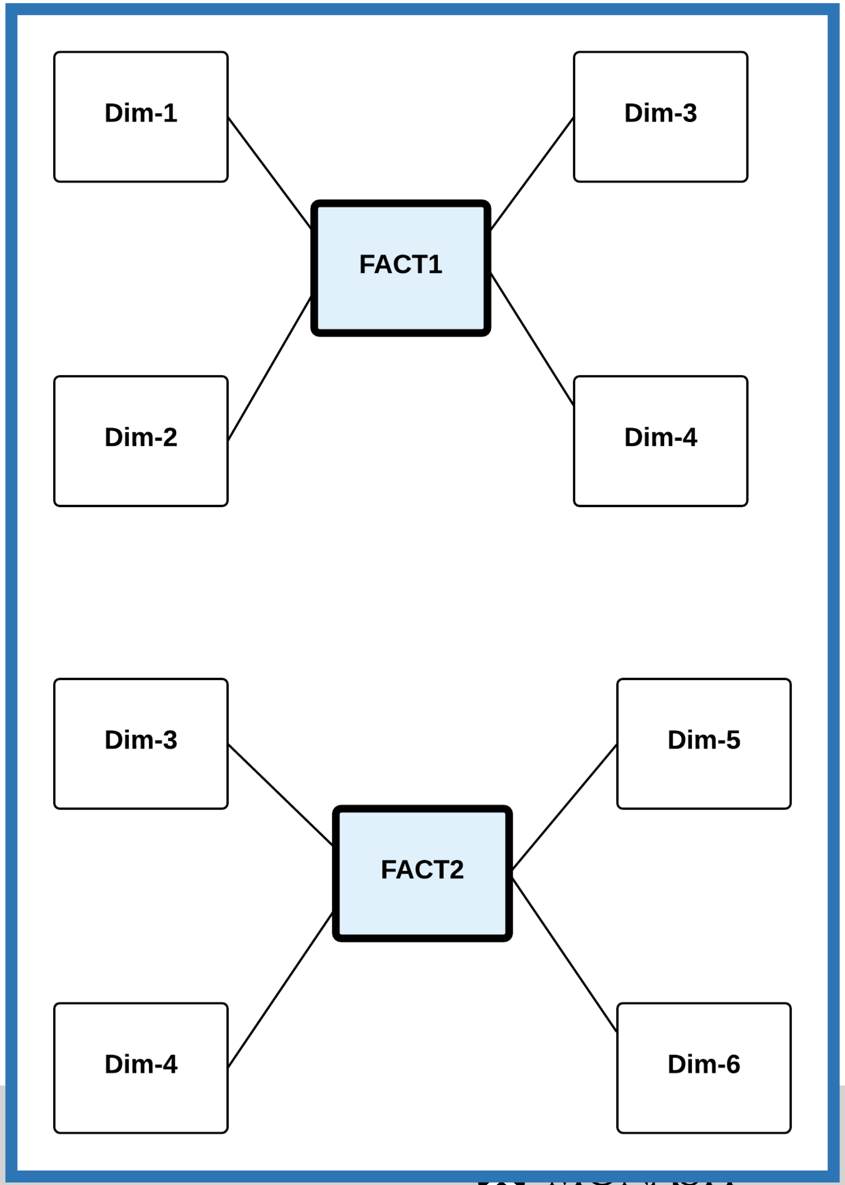
- A *Subject-Oriented* data warehouse means that one star schema focuses on one subject only.
    - A subject refers to a topic of analysis.
- Example:
    - Sales of properties & Rental properties
        - The two star schemas should not be combined since both focus on different subjects.
        - But two separate star schemas can still share the same dimensions



# Multi-Fact



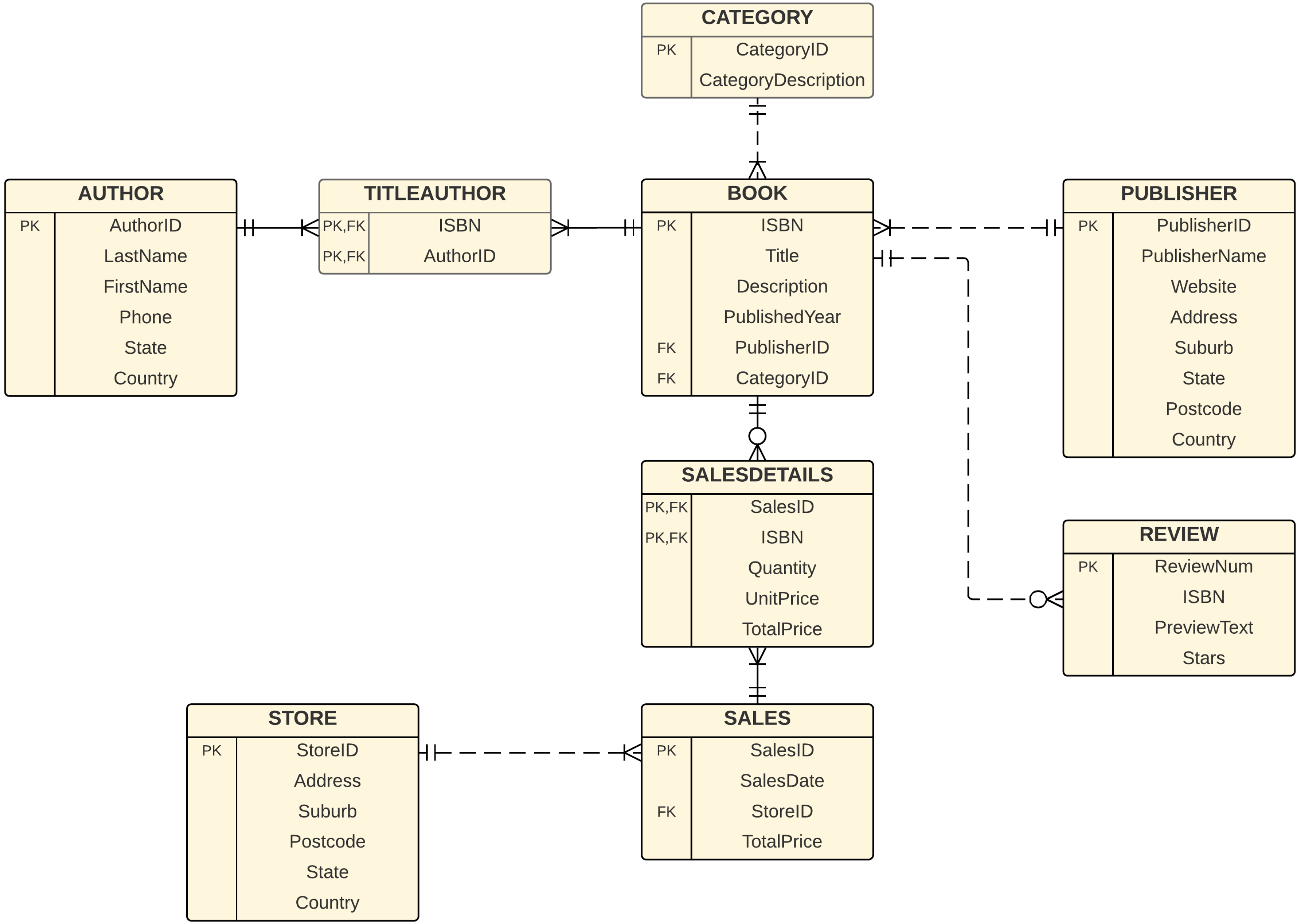
VS



# Different Subject Multi-Fact: The Book Sales Case Study

MONASH University

# The Book Sales Case Study



# The Book Sales Case Study

The system stores information about books, including the authors, publishers, book categories, as well as the reviews that each book has received. The ”stars” attribute in the Review entity records the star rating for each review (e.g. 5 stars for excellent to 1 star for poor, etc). One book may receive many reviews. For simplicity, it is assumed, as also shown in the E/R diagram, that a book will only have one category.

The E/R diagram also includes entities related to sales of books, and the stores which sale the books. Each store has many sales transactions (i.e. the Sales entity), and each sales transaction may include several books (i.e. the SalesDetails entity). The Total Price attribute in the Sales Details entity is basically Quantity multiplied by the Unit Price, whereas Total Price in the Sales entity is the total price for each sales transaction.

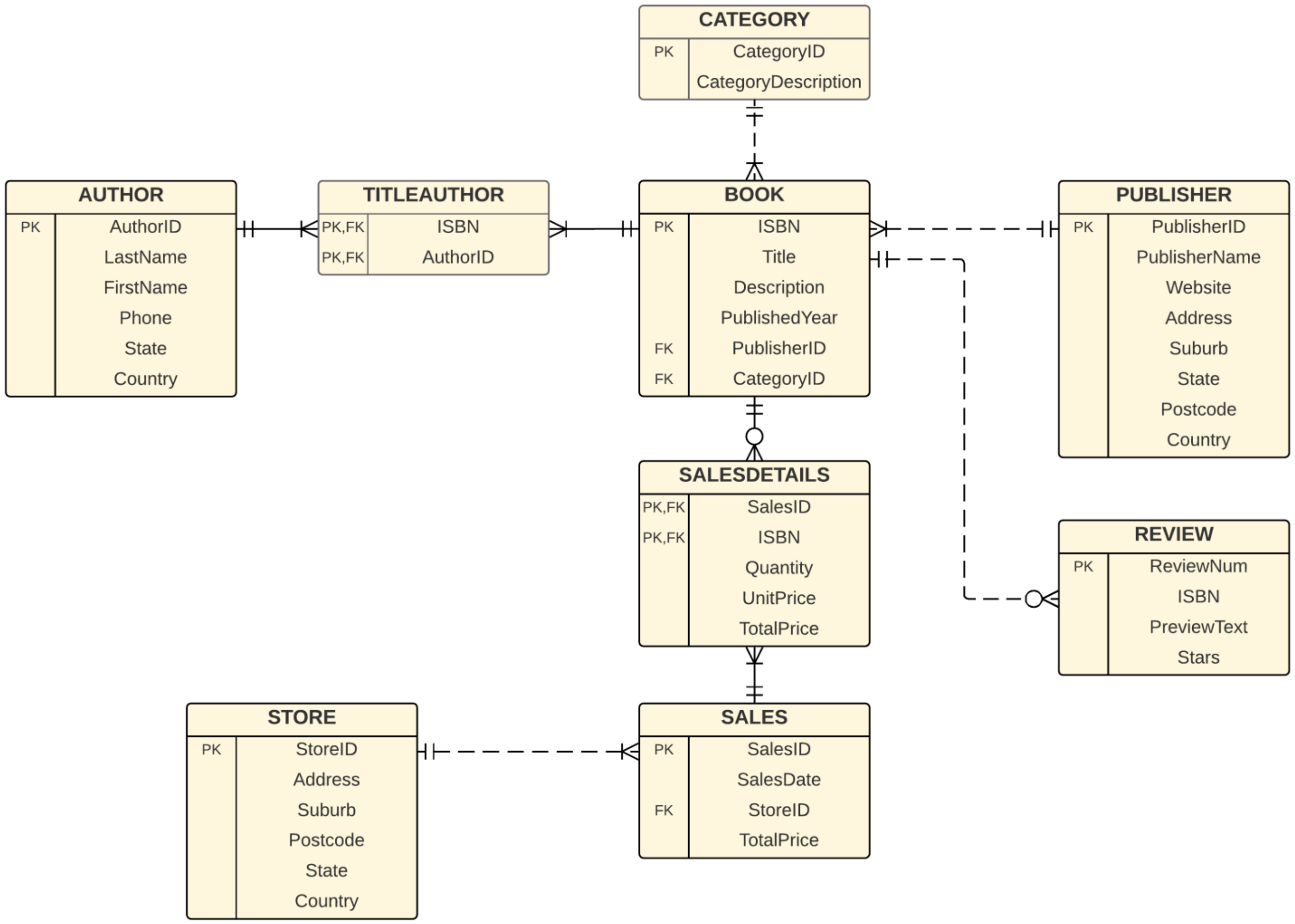
# The Book Sales Case Study

The requirements for the data warehouse are quite simple. The data warehouse must be able to answer at least the following questions:

- What are the total sales for each bookstore in a month?
- What is the number of books sold for each category?
- What is the book category that has the highest total sales?
- What is the number of reviews for each category?
- How many 5-star reviews for each category?

# The Book Sales Case Study

- Based on the requirements, it is clear that there are three fact measures:
  1. Total sales,
  2. Number of books sold, and
  3. Number of reviews.



**CATEGORY**

| | |
|---|---|
| PK | CategoryID |
| | CategoryDescription |

**AUTHOR**

| | |
|---|---|
| PK | AuthorID |
| | LastName |
| | FirstName |
| | Phone |
| | State |
| | Country |

**TITLEAUTHOR**

| | |
|---|---|
| PK,FK | ISBN |
| PK,FK | AuthorID |

**BOOK**

| | |
|---|---|
| PK | ISBN |
| | Title |
| | Description |
| | PublishedYear |
| FK | PublisherID |
| FK | CategoryID |

**PUBLISHER**

| | |
|---|---|
| PK | PublisherID |
| | PublisherName |
| | Website |
| | Address |
| | Suburb |
| | State |
| | Postcode |
| | Country |

**SALESDETAILS**

| | |
|---|---|
| PK,FK | SalesID |
| PK,FK | ISBN |
| | Quantity |
| | UnitPrice |
| | TotalPrice |

**REVIEW**

| | |
|---|---|
| PK | ReviewNum |
| | ISBN |
| | PreviewText |
| | Stars |

**STORE**

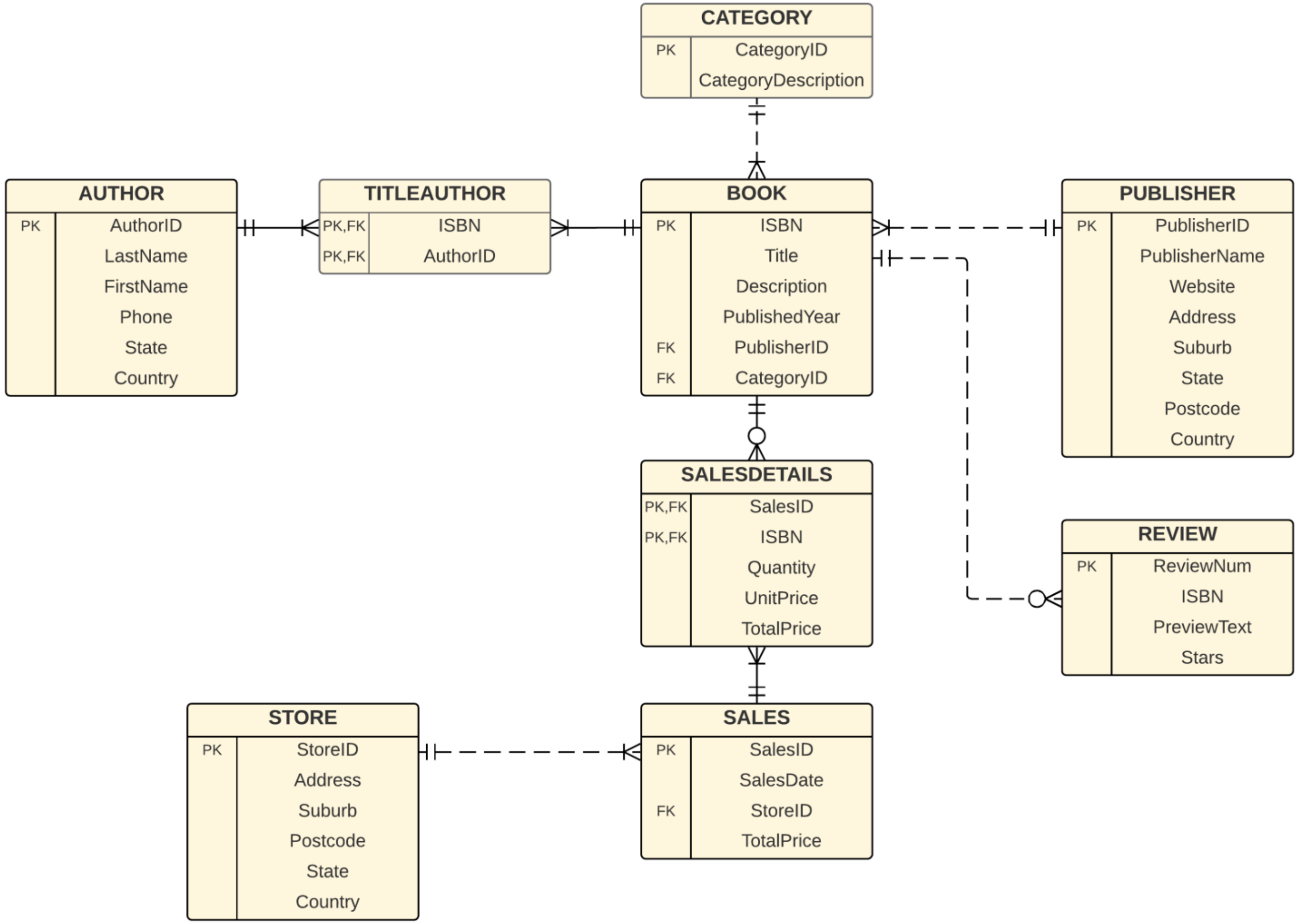
| | |
|---|---|
| PK | StoreID |
| | Address |
| | Suburb |
| | Postcode |
| | State |
| | Country |

**SALES**

| | |
|---|---|
| PK | SalesID |
| | SalesDate |
| FK | StoreID |
| | TotalPrice |

# The Book Sales Case Study

- Based on the requirements, it is clear that there are three fact measures:
  1. Total sales,
  2. Number of books sold, and
  3. Number of reviews.
- The dimensions:
  1. Store dimension,
  2. Time dimension,
  3. Book category dimension, and
  4. Star rating dimension.



| | CATEGORY |
|---|---|
| PK | CategoryID |
| | CategoryDescription |

| | AUTHOR |
|---|---|
| PK | AuthorID |
| | LastName |
| | FirstName |
| | Phone |
| | State |
| | Country |

| | TITLEAUTHOR |
|---|---|
| PK,FK | ISBN |
| PK,FK | AuthorID |

| | BOOK |
|---|---|
| PK | ISBN |
| | Title |
| | Description |
| | PublishedYear |
| FK | PublisherID |
| FK | CategoryID |

| | PUBLISHER |
|---|---|
| PK | PublisherID |
| | PublisherName |
| | Website |
| | Address |
| | Suburb |
| | State |
| | Postcode |
| | Country |

| | SALESDETAILS |
|---|---|
| PK,FK | SalesID |
| PK,FK | ISBN |
| | Quantity |
| | UnitPrice |
| | TotalPrice |

| | REVIEW |
|---|---|
| PK | ReviewNum |
| | ISBN |
| | PreviewText |
| | Stars |

| | STORE |
|---|---|
| PK | StoreID |
| | Address |
| | Suburb |
| | Postcode |
| | State |
| | Country |

| | SALES |
|---|---|
| PK | SalesID |
| | SalesDate |
| FK | StoreID |
| | TotalPrice |

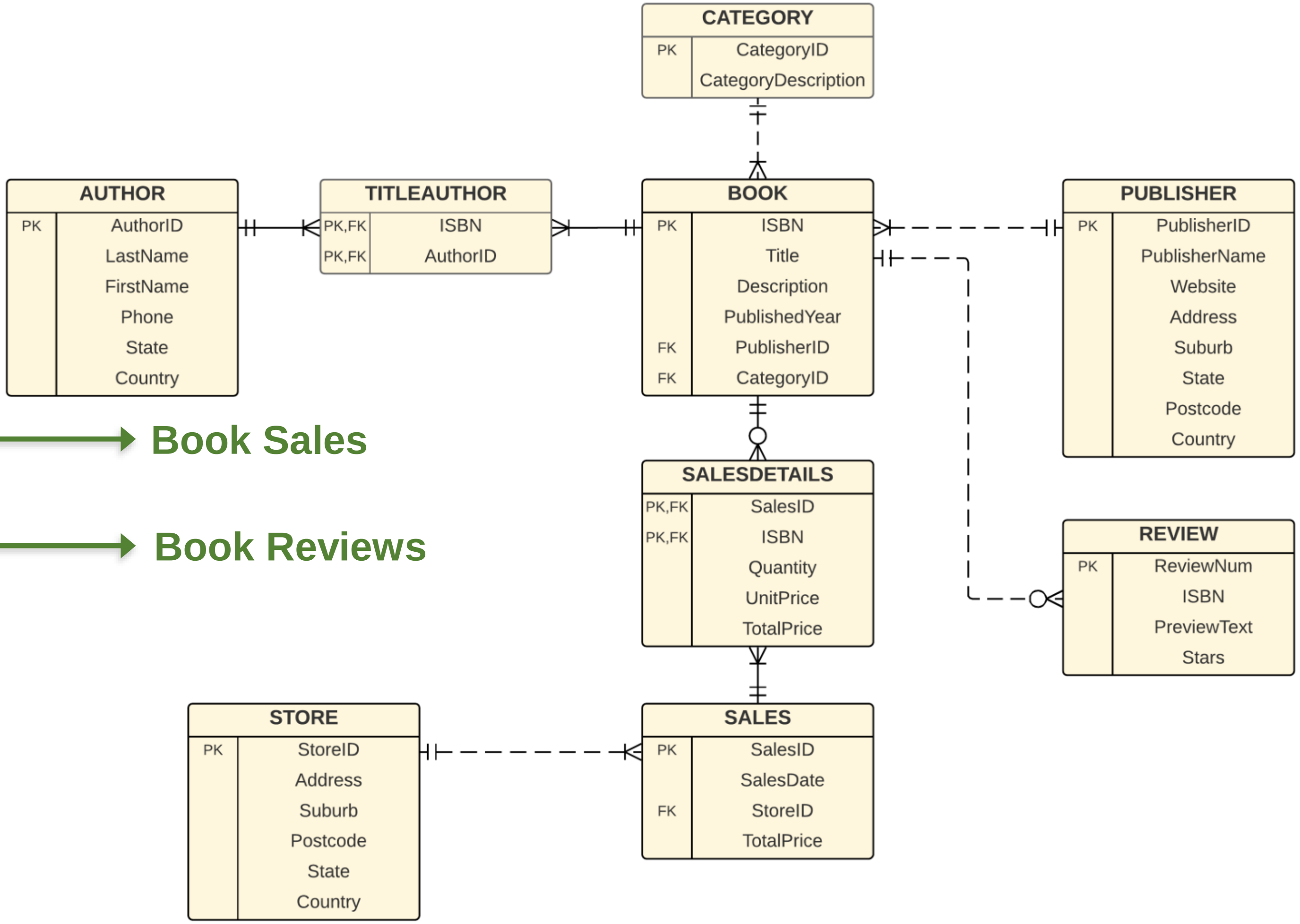
# The Book Sales Case Study

- Based on the requirements, it is clear that there are three fact measures:
  1. Total sales,
  2. Number of books sold, and
  3. Number of reviews.

Book Sales

Book Reviews

- The dimensions:
  1. Store dimension,
  2. Time dimension,
  3. Book category dimension, and
  4. Star rating dimension.



| CATEGORY | |
|---|---|
| PK | CategoryID |
| | CategoryDescription |

| AUTHOR | |
|---|---|
| PK | AuthorID |
| | LastName |
| | FirstName |
| | Phone |
| | State |
| | Country |

| TITLEAUTHOR | |
|---|---|
| PK,FK | ISBN |
| PK,FK | AuthorID |

| BOOK | |
|---|---|
| PK | ISBN |
| | Title |
| | Description |
| | PublishedYear |
| FK | PublisherID |
| FK | CategoryID |

| PUBLISHER | |
|---|---|
| PK | PublisherID |
| | PublisherName |
| | Website |
| | Address |
| | Suburb |
| | State |
| | Postcode |
| | Country |

| SALESDETAILS | |
|---|---|
| PK,FK | SalesID |
| PK,FK | ISBN |
| | Quantity |
| | UnitPrice |
| | TotalPrice |

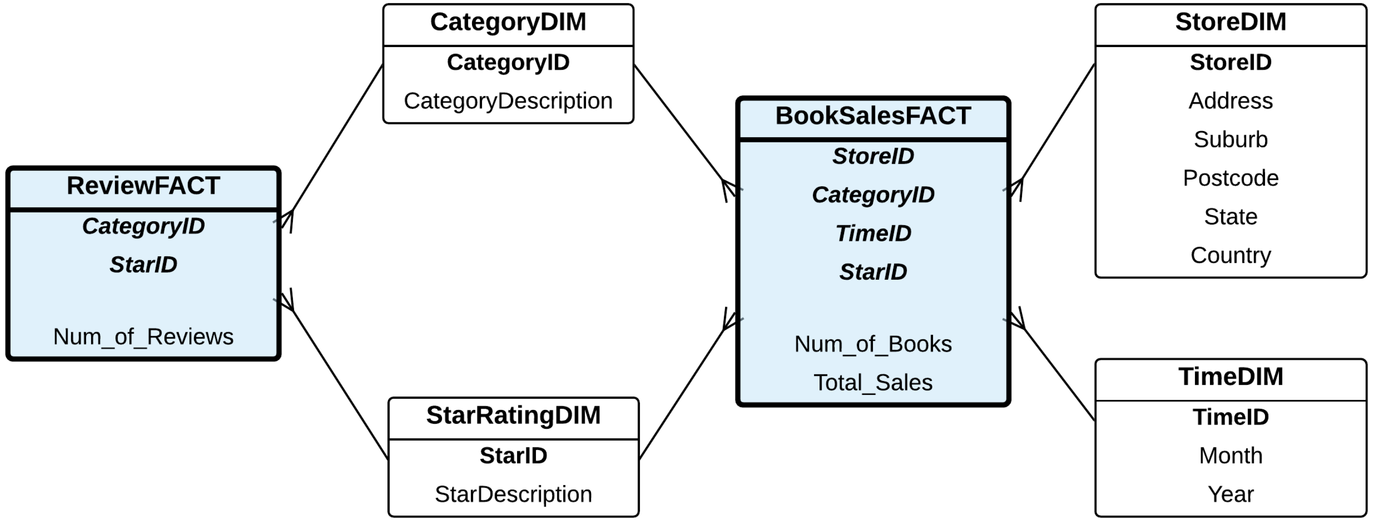
| REVIEW | |
|---|---|
| PK | ReviewNum |
| | ISBN |
| | PreviewText |
| | Stars |

| STORE | |
|---|---|
| PK | StoreID |
| | Address |
| | Suburb |
| | Postcode |
| | State |
| | Country |

| SALES | |
|---|---|
| PK | SalesID |
| | SalesDate |
| FK | StoreID |
| | TotalPrice |

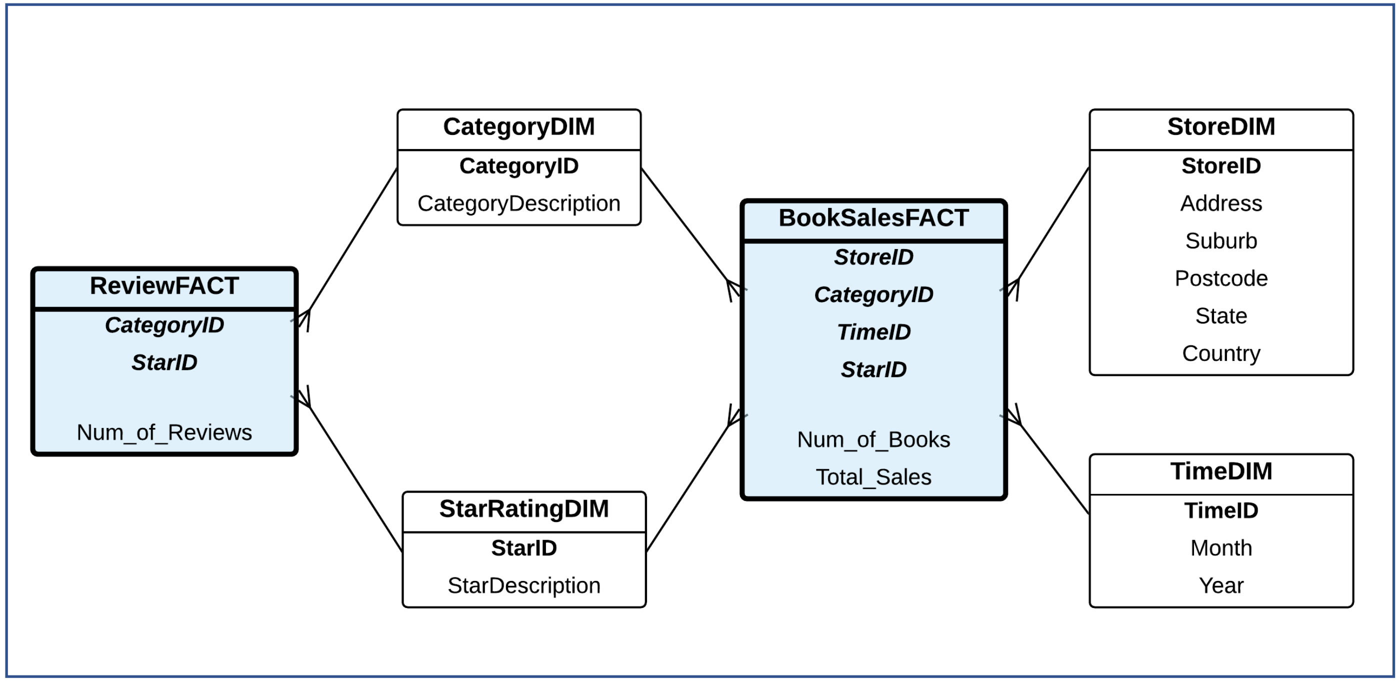
# The Book Sales Case Study

- Based on the requirements, it is clear that there are three fact measures:
  1. Total sales,
  2. Number of books sold, and
  3. Number of reviews.
- The dimensions:
  1. Store dimension,
  2. Time dimension,
  3. Book category dimension, and
  4. Star rating dimension.



# The Book Sales Case Study

## (a) A Multi-Fact Star Schema



ReviewFACT
*CategoryID*
*StarID*
Num_of_Reviews

CategoryDIM
**CategoryID**
CategoryDescription

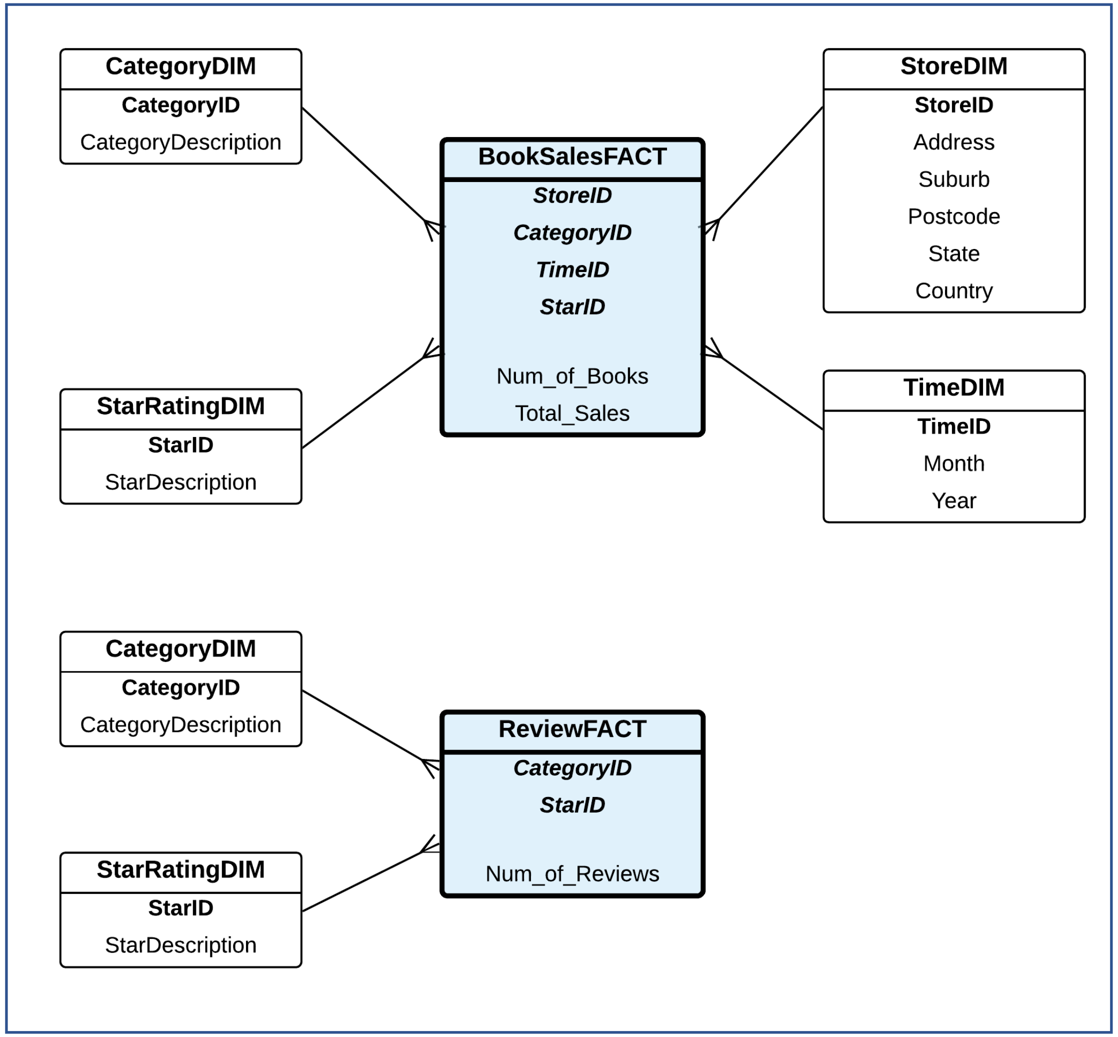
StarRatingDIM
**StarID**
StarDescription

BookSalesFACT
*StoreID*
*CategoryID*
*TimeID*
*StarID*
Num_of_Books
Total_Sales

StoreDIM
**StoreID**
Address
Suburb
Postcode
State
Country

TimeDIM
**TimeID**
Month
Year

## (b) Two Star Schemas



CategoryDIM
**CategoryID**
CategoryDescription

BookSalesFACT
*StoreID*
*CategoryID*
*TimeID*
*StarID*
Num_of_Books
Total_Sales

StoreDIM
**StoreID**
Address
Suburb
Postcode
State
Country

StarRatingDIM
**StarID**
StarDescription

TimeDIM
**TimeID**
Month
Year

CategoryDIM
**CategoryID**
CategoryDescription

ReviewFACT
*CategoryID*
*StarID*
Num_of_Reviews

StarRatingDIM
**StarID**
StarDescription

# The Book Sales Case Study

- To create Category Dimension:

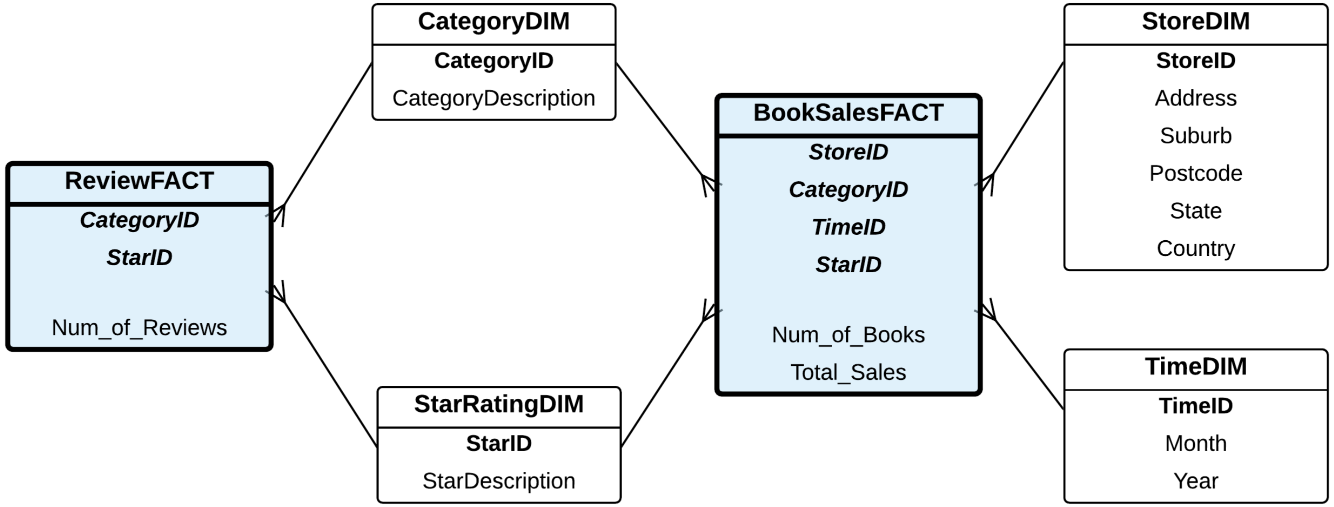
```
- create table CategoryDim
   as select * from Category;
```

- To create Store Dimension:

```
- create table StoreDim
   as select * from Store;
```

- To create Time Dimension:

```
- create table TimeDim
   as select distinct
      to_char(SalesDate, 'YYYYMM') as TimeID,
      to_char(SalesDate, 'MM') as Month,
      to_char(SalesDate, 'YYYY') as Year
   from Sales;
```



# The Book Sales Case Study

- To create Star Rating Dimension:

  -
    ```
    create table StarRatingDim
     (StarID number(1),
      StarDescription varchar2(15));
    ```

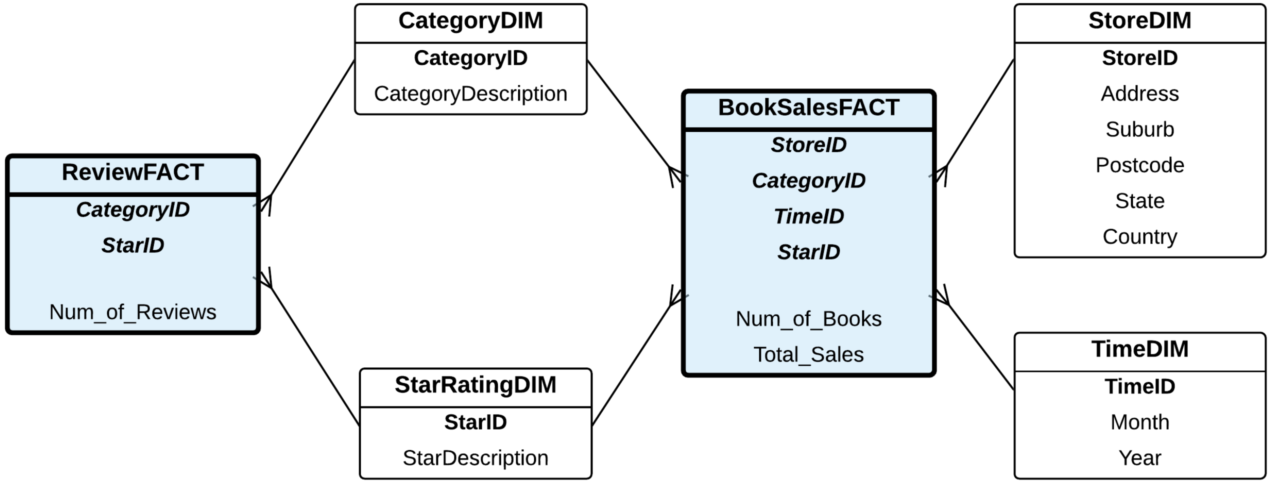
- Inserting values to Star Rating Dimension:

  -
    ```
    insert into StarRatingDim values (0, ’Unknown’);
    insert into StarRatingDim values (1, ’Poor’);
    insert into StarRatingDim values (2, ’Not Good’);
    insert into StarRatingDim values (3, ’Average’);
    insert into StarRatingDim values (4, ’Good’);
    insert into StarRatingDim values (5, ’Excellent’);
    ```

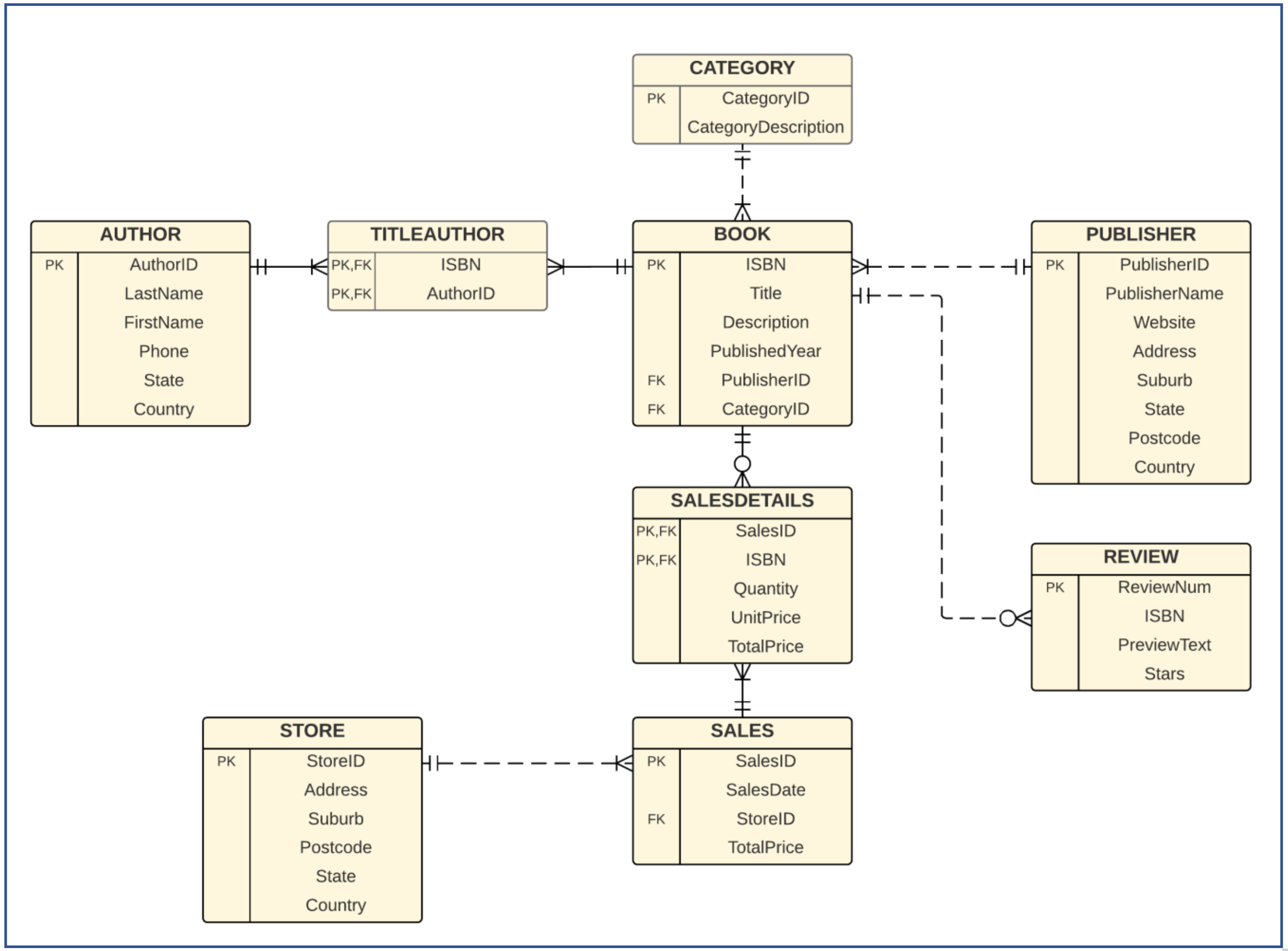
# The Book Sales Case Study

- To create Review Fact:

```
- create table ReviewFact as
   select
      B.CategoryID,
      R.Stars as StarID,
      count(*) as Num_of_Reviews
   from Book5 B, Review5 R
   where B.ISBN=R.ISBN
   group by B.CategoryID, R.Stars;
```



# The Book Sales Case Study



**CATEGORY**

| | |
|---|---|
| PK | CategoryID |
| | CategoryDescription |

**AUTHOR**

| | |
|---|---|
| PK | AuthorID |
| | LastName |
| | FirstName |
| | Phone |
| | State |
| | Country |

**TITLEAUTHOR**

| | |
|---|---|
| PK,FK | ISBN |
| PK,FK | AuthorID |

**BOOK**

| | |
|---|---|
| PK | ISBN |
| | Title |
| | Description |
| | PublishedYear |
| FK | PublisherID |
| FK | CategoryID |

**PUBLISHER**

| | |
|---|---|
| PK | PublisherID |
| | PublisherName |
| | Website |
| | Address |
| | Suburb |
| | State |
| | Postcode |
| | Country |

**SALESDETAILS**

| | |
|---|---|
| PK,FK | SalesID |
| PK,FK | ISBN |
| | Quantity |
| | UnitPrice |
| | TotalPrice |

**REVIEW**

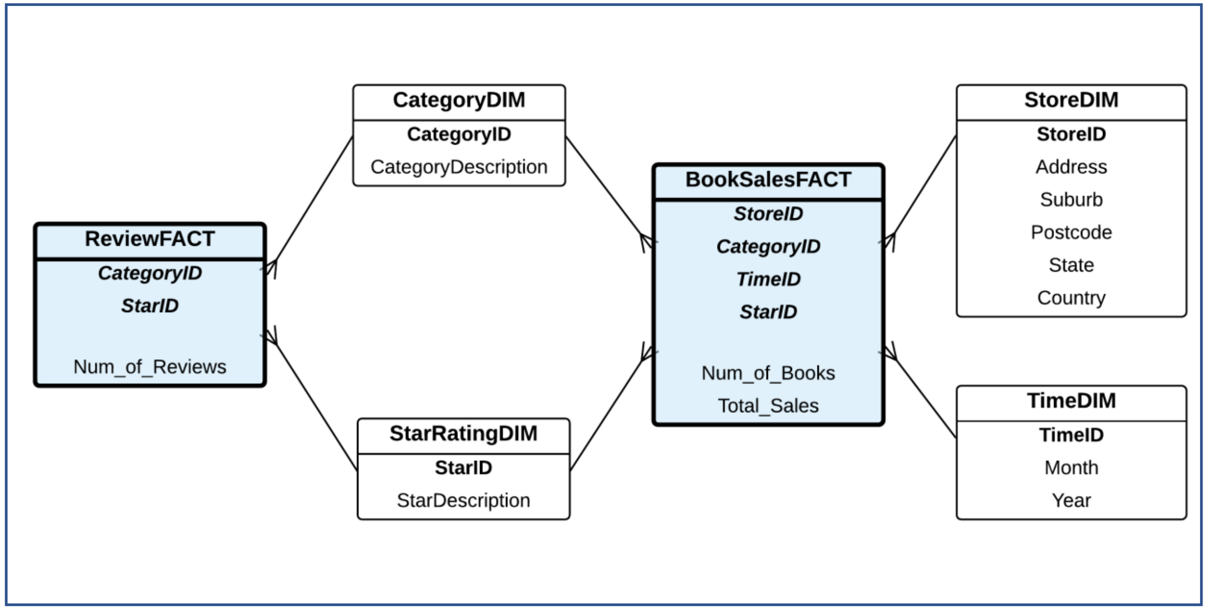
| | |
|---|---|
| PK | ReviewNum |
| | ISBN |
| | PreviewText |
| | Stars |

**STORE**

| | |
|---|---|
| PK | StoreID |
| | Address |
| | Suburb |
| | Postcode |
| | State |
| | Country |

**SALES**

| | |
|---|---|
| PK | SalesID |
| | SalesDate |
| FK | StoreID |
| | TotalPrice |



**ReviewFACT**

***CategoryID***
***StarID***

Num_of_Reviews

**CategoryDIM**

**CategoryID**
CategoryDescription

**StarRatingDIM**

**StarID**
StarDescription

**BookSalesFACT**

***StoreID***
***CategoryID***
***TimeID***
***StarID***

Num_of_Books
Total_Sales

**StoreDIM**

**StoreID**
Address
Suburb
Postcode
State
Country

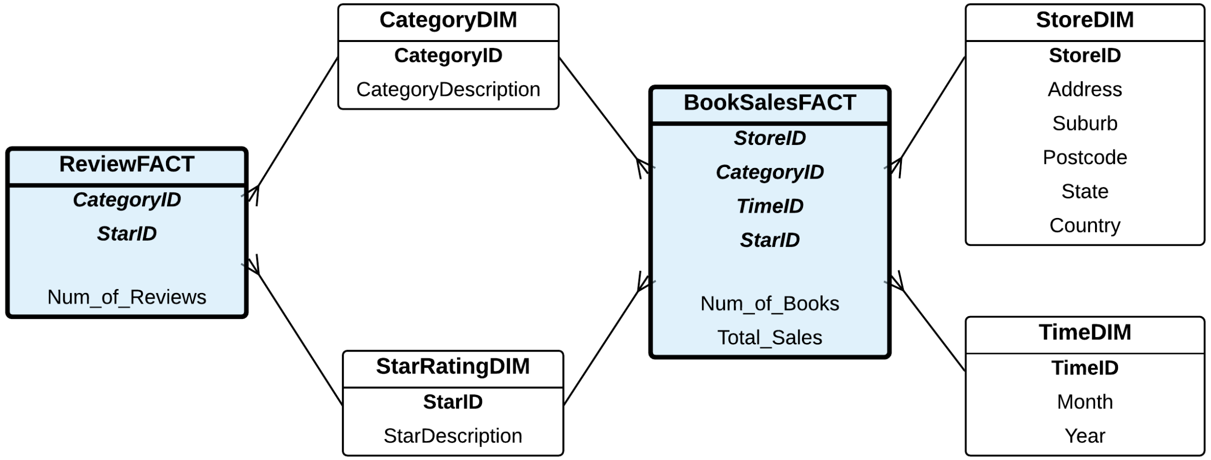
**TimeDIM**

**TimeID**
Month
Year

# The Book Sales Case Study

- To do an outer join between Book and Review:

```
- create table TempBookWithStar as
  select
     B.ISBN,
     B.CategoryID,
     nvl(R.Stars, 0) as Stars
  from Book5 B, Review5 R
  where B.ISBN = R.ISBN(+);
```



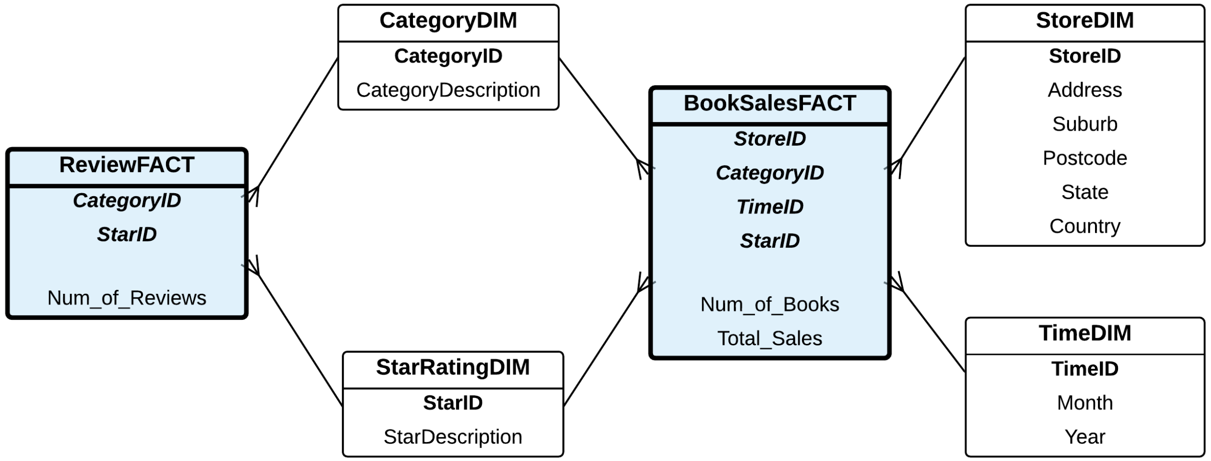
- To calculate average stars:

```
- create table TempBookWithAvgStar as
  select ISBN, CategoryID, round(avg(Stars)) as Avg_Stars
  from TempBookWithStar
  group by ISBN, CategoryID;
```

# The Book Sales Case Study

- To do an outer join between Book and Review:

```
– create table TempBookWithStar as
  select
     B.ISBN,
     B.CategoryID,
     nvl(R.Stars, 0) as Stars
  from Book5 B, Review5 R
  where B.ISBN = R.ISBN(+);
```



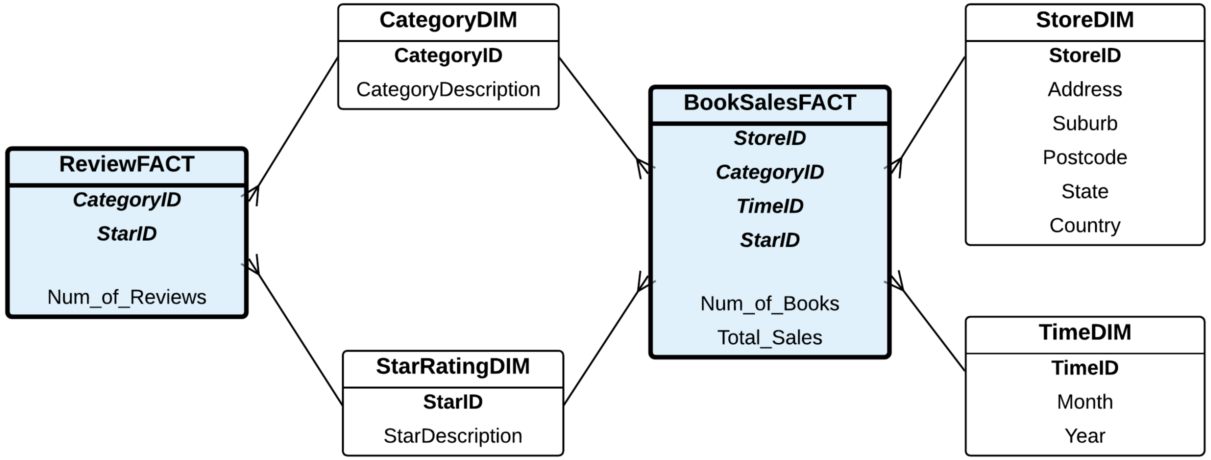
- To calculate average stars:

```
– create table TempBookWithAvgStar as
  select ISBN, CategoryID, round(avg(Stars)) as Avg_Stars
  from TempBookWithStar
  group by ISBN, CategoryID;
```

# The Book Sales Case Study

- To do an outer join between Book and Review:

```
- create table TempBookWithStar as
  select
     B.ISBN,
     B.CategoryID,
     nvl(R.Stars, 0) as Stars
  from Book5 B, Review5 R
  where B.ISBN = R.ISBN(+);
```



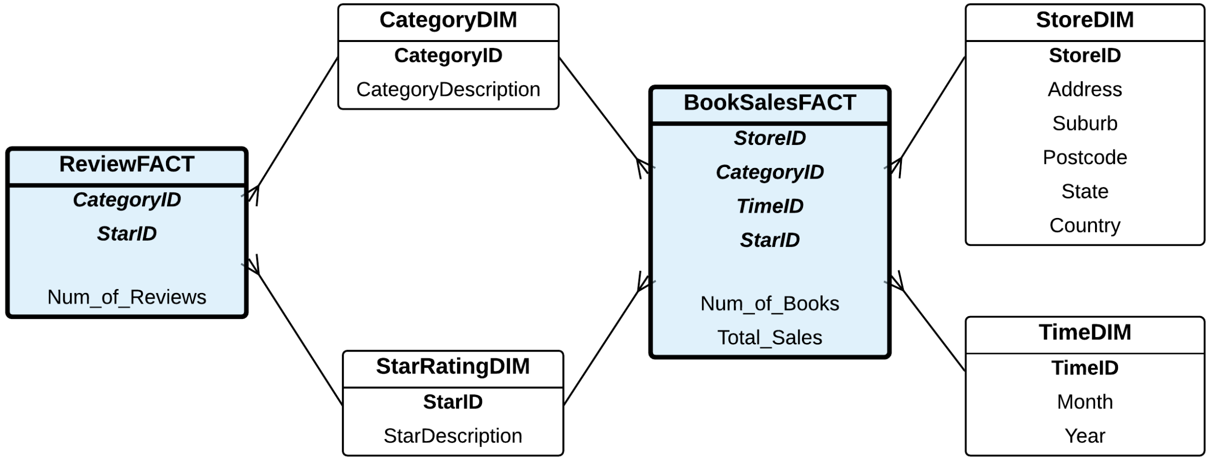
- To calculate average stars:

```
- create table TempBookWithAvgStar as
  select ISBN, CategoryID, round(avg(Stars)) as Avg_Stars
  from TempBookWithStar
  group by ISBN, CategoryID;
```

# The Book Sales Case Study

- To do an outer join between Book and Review:

  -
    ```
    create table TempBookWithStar as
     select
        B.ISBN,
        B.CategoryID,
        nvl(R.Stars, 0) as Stars
     from Book5 B, Review5 R
     where B.ISBN = R.ISBN(+);
    ```



**ReviewFACT**
*CategoryID*
*StarID*
Num_of_Reviews

**CategoryDIM**
**CategoryID**
CategoryDescription

**StarRatingDIM**
**StarID**
StarDescription

**BookSalesFACT**
*StoreID*
*CategoryID*
*TimeID*
*StarID*
Num_of_Books
Total_Sales

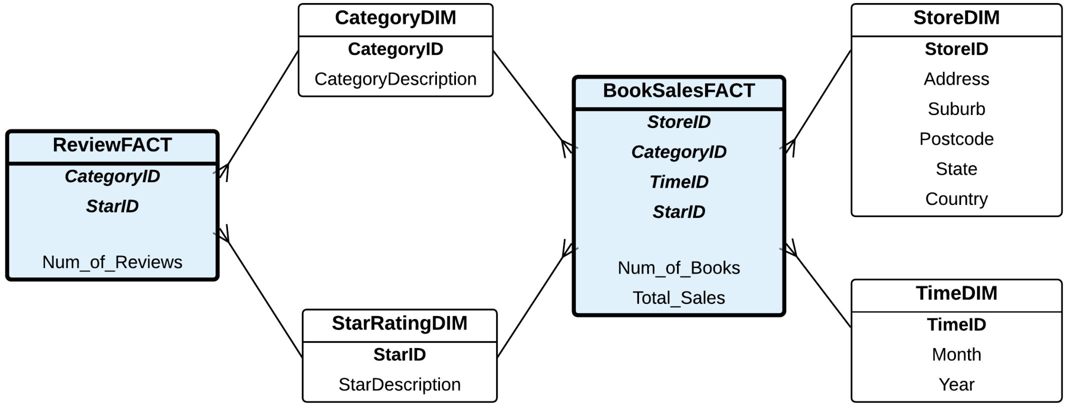
**StoreDIM**
**StoreID**
Address
Suburb
Postcode
State
Country

**TimeDIM**
**TimeID**
Month
Year

- To calculate average stars:

  -
    ```
    create table TempBookWithAvgStar as
     select ISBN, CategoryID, round(avg(Stars)) as Avg_Stars
     from TempBookWithStar
     group by ISBN, CategoryID;
    ```

# The Book Sales Case Study



CategoryDIM
**CategoryID**
CategoryDescription

StoreDIM
**StoreID**
Address
Suburb
Postcode
State
Country

ReviewFACT
***CategoryID***
***StarID***
Num_of_Reviews

BookSalesFACT
***StoreID***
***CategoryID***
***TimeID***
***StarID***
Num_of_Books
Total_Sales

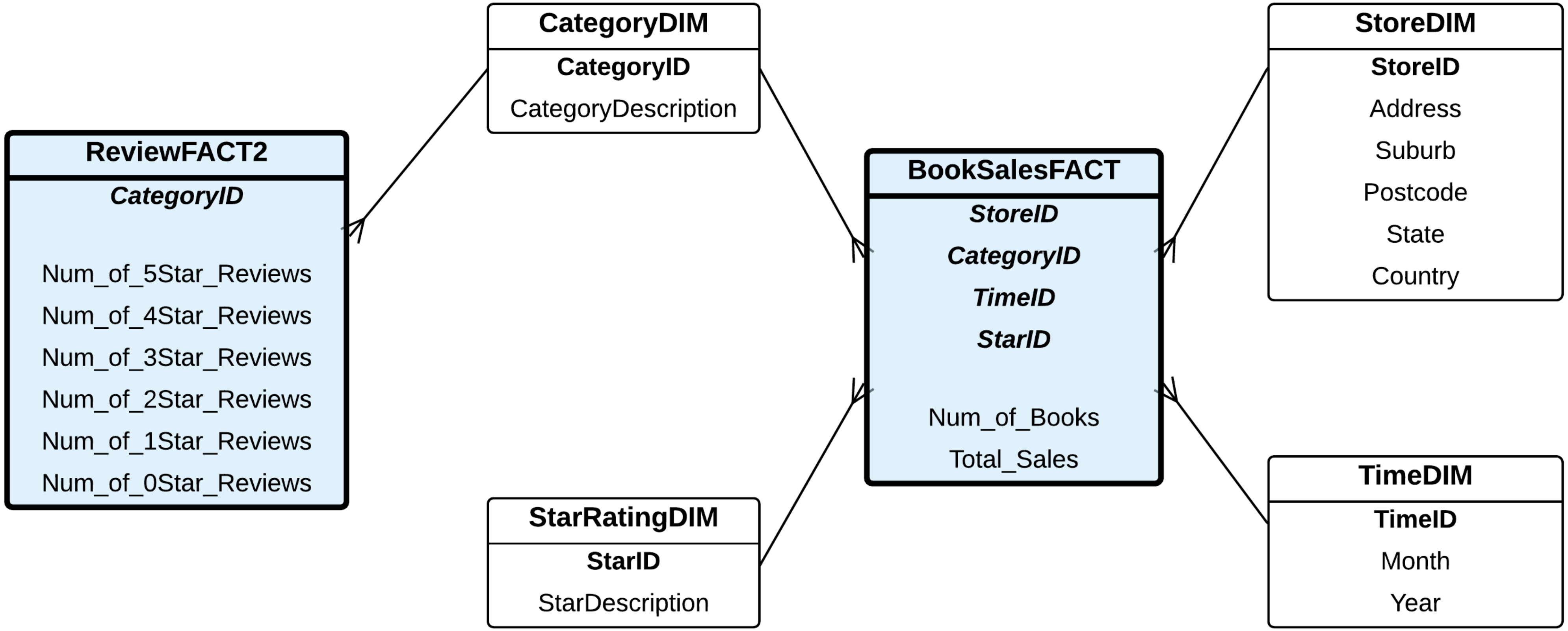
StarRatingDIM
**StarID**
StarDescription

TimeDIM
**TimeID**
Month
Year

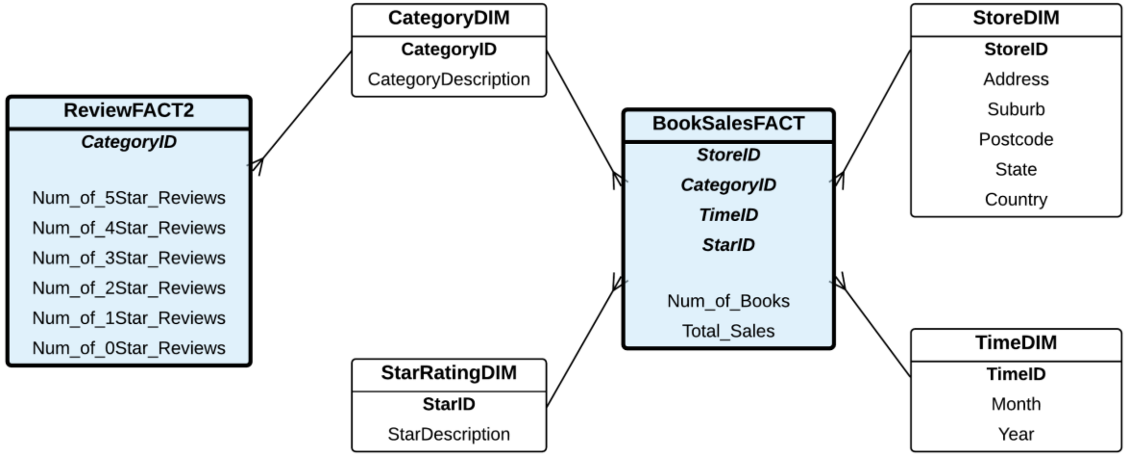
- To create Book Sales Fact:

```
- create table BookSalesFact as
  select T.CategoryID,
         to_char(S.SalesDate, ’YYYYMM’) as TimeID,
         S.StoreID,
         T.Avg_Stars as StarID,
         sum(D.Quantity) as Num_of_Books,
         sum(D.TotalPrice) as Total_Sales
  from TempBookWithAvgStar T, Sales5 S, SalesDetails5 D
  where T.ISBN = D.ISBN
  and D.SalesID = D.SalesID
  group by T.CategoryID, to_char(S.SalesDate, ’YYYYMM’), S.StoreID,
           T.Avg_Stars;
```

# Multi-Fact with Pivot Table



# Multi-Fact with Pivot Table



- To create Review Fact with Pivot Table:
  - `create table ReviewFact2 as`

```
select CategoryID,
       0 as Num_of_0Star_Reviews,
       0 as Num_of_1Star_Reviews,
       0 as Num_of_2Star_Reviews,
       0 as Num_of_3Star_Reviews,
       0 as Num_of_4Star_Reviews, 0 as Num_of_5Star_Reviews
from CategoryDim;
```

# Multi-Fact with Pivot Table

- Update rating value in Review Fact with Pivot Table:

  -
    ```
    update ReviewFact2 F2
    set
      Num_of_1Star_Reviews =
        nvl((select Num_of_Reviews
          from ReviewFact F1
          where F2.CategoryID = F1.CategoryID
          and F1.StarID = 1),0),
      Num_of_2Star_Reviews =
        nvl((select Num_of_Reviews
          from ReviewFact F1
          where F2.CategoryID = F1.CategoryID
          and F1.StarID = 2),0),
      Num_of_3Star_Reviews =
        nvl((select Num_of_Reviews
          from ReviewFact F1
          where F2.CategoryID = F1.CategoryID
          and F1.StarID = 3),0),
      Num_of_4Star_Reviews =
        nvl((select Num_of_Reviews
          from ReviewFact F1
          where F2.CategoryID = F1.CategoryID
          and F1.StarID = 4),0),
      Num_of_5Star_Reviews =
        nvl((select Num_of_Reviews
          from ReviewFact F1
          where F2.CategoryID = F1.CategoryID
          and F1.StarID = 5),0);
    ```

# Multi-Fact or Single Fact with multiple Fact Measures: A Private Taxi Company Case Study

MONASH
University

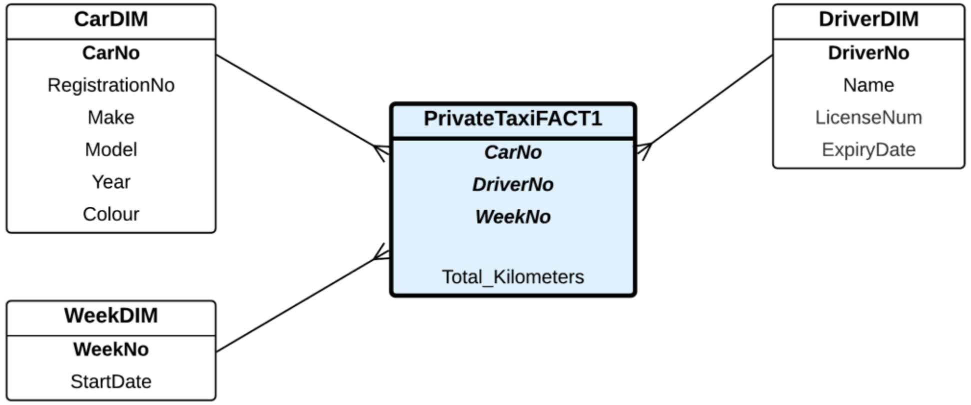
# A Private Taxi Company Case Study

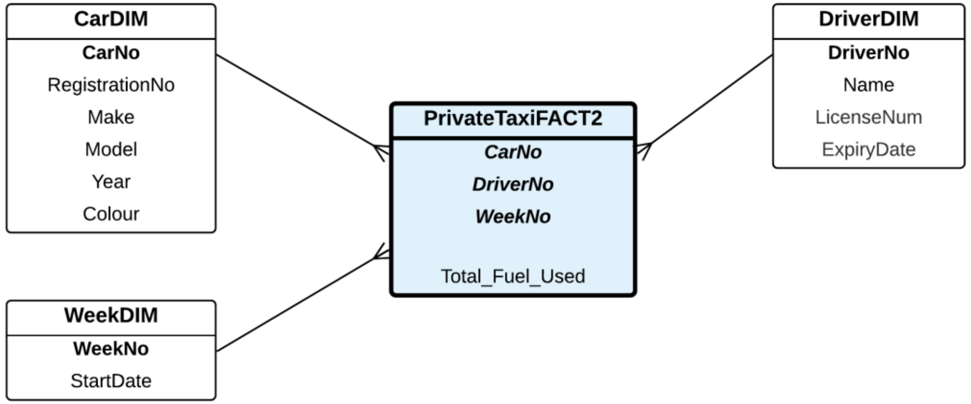
A private taxi company is a small business. The owner of this company owns five cars, which can be chartered like a taxi. The business has many clients, and the number of clients grow due to word of mouth. When a client needs a service from this company (such as sending them from their home to the airport, or vice versa), they send a booking though an SMS or a chatting social media (e.g. WhatsApp). Once the booking is confirmed, a driver will pick the clients up and send them to the destination. The charges are normally a pre-defined fixed price, such as from an inner suburb of the city to the airport is $90. The charge might be slightly more expensive than the normal taxi, but somehow many clients prefer the services that this company offer.

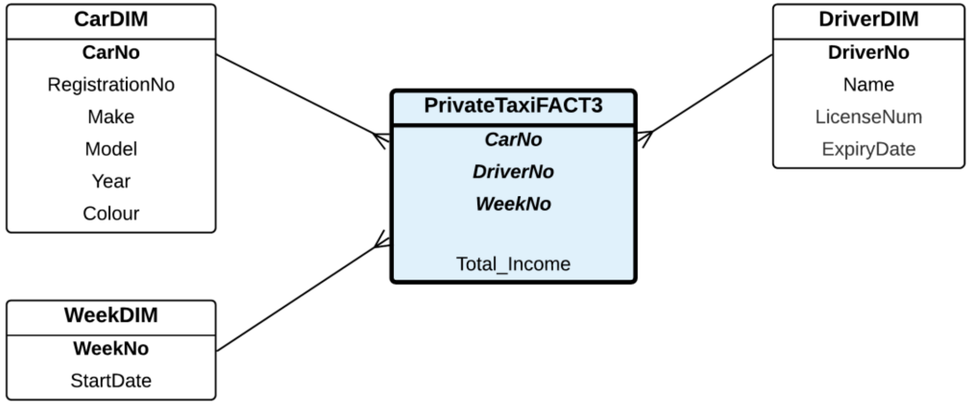
The company has three full-time drivers, including the owner himself, and five sessional or part-time drivers who may be called when the company needs drivers. Every time a client hires this taxi, the information (e.g. origin and destination, pickup time and date, fares, driver, car, distance travelled, amount of petrol used, etc) is recorded into their operational database.

# A Private Taxi Company Case

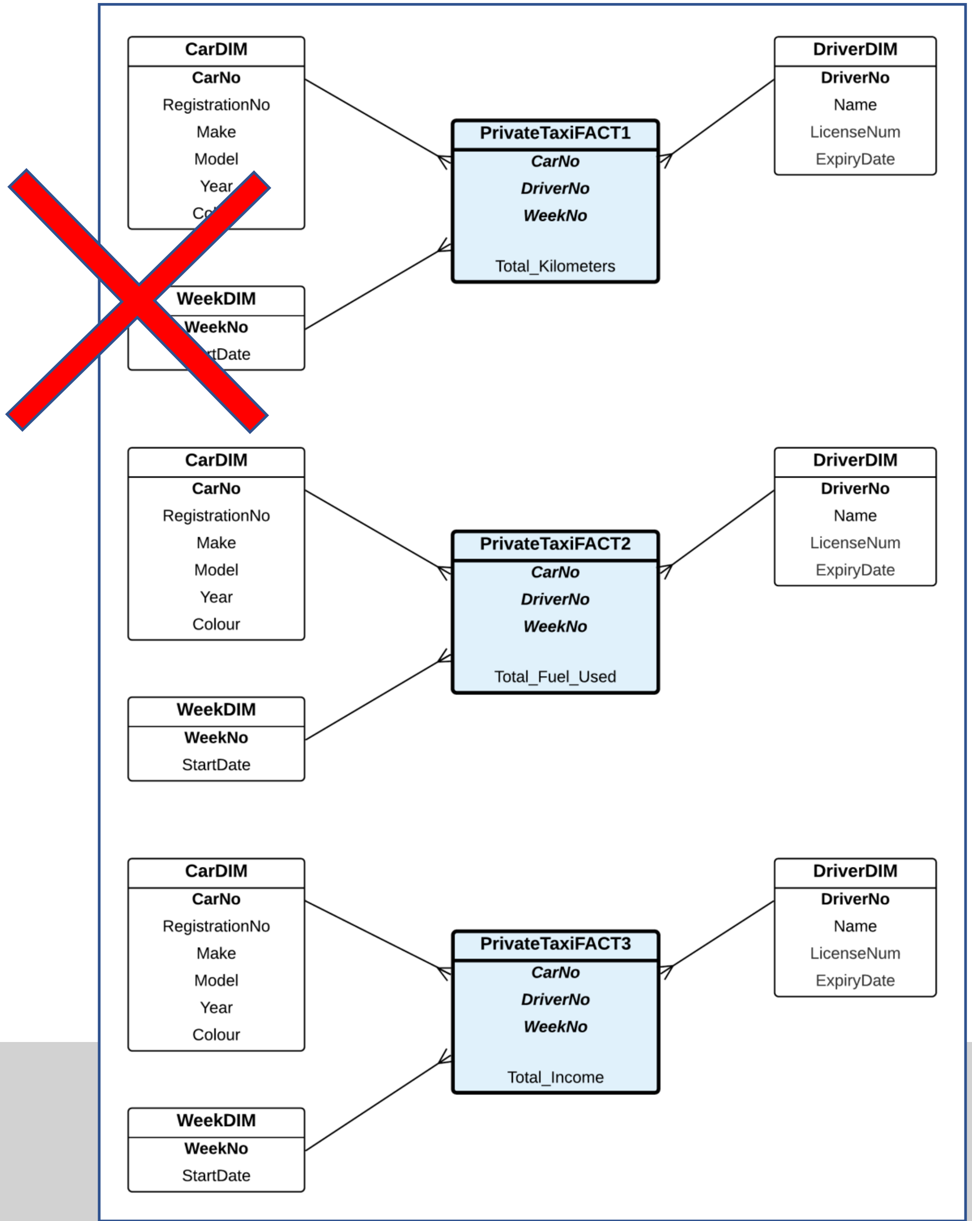
On top of this operational database, the company also builds a data warehouse for reporting purposes. The data warehouse is a star schema with only three dimensions: Car dimension, Driver dimension, and Week dimension. Basically the company would like to analyze their total income, total kilometer travelled, and total fuel used. These then become the fact measures of the star schema.







# A Private Taxi Company Case Study



CarDIM
CarNo
RegistrationNo
Make
Model
Year
Co

WeekDIM
WeekNo
rtDate

PrivateTaxiFACT1
*CarNo*
*DriverNo*
*WeekNo*
Total_Kilometers

DriverDIM
DriverNo
Name
LicenseNum
ExpiryDate

CarDIM
CarNo
RegistrationNo
Make
Model
Year
Colour

WeekDIM
WeekNo
StartDate

PrivateTaxiFACT2
*CarNo*
*DriverNo*
*WeekNo*
Total_Fuel_Used

DriverDIM
DriverNo
Name
LicenseNum
ExpiryDate

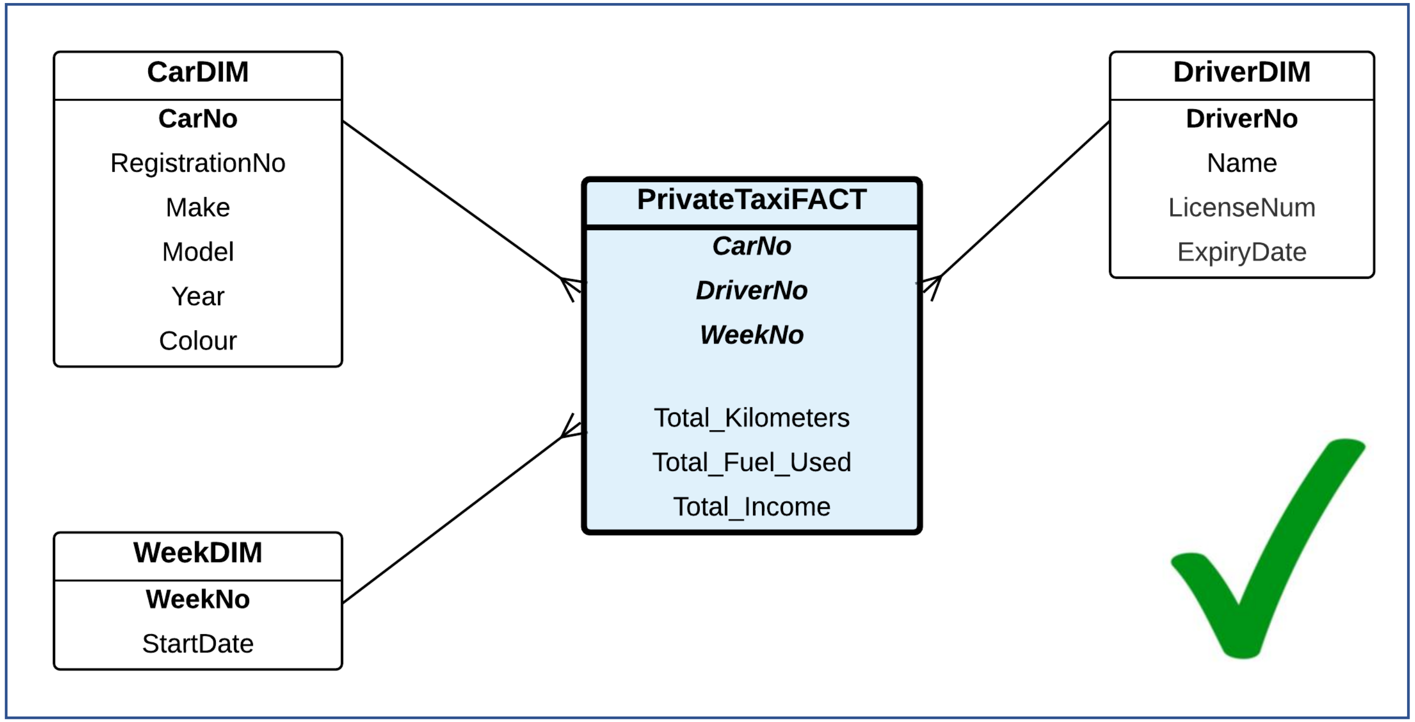
CarDIM
CarNo
RegistrationNo
Make
Model
Year
Colour

WeekDIM
WeekNo
StartDate

PrivateTaxiFACT3
*CarNo*
*DriverNo*
*WeekNo*
Total_Income

DriverDIM
DriverNo
Name
LicenseNum
ExpiryDate

## Just merge all dim into one



CarDIM
CarNo
RegistrationNo
Make
Model
Year
Colour

WeekDIM
WeekNo
StartDate

PrivateTaxiFACT
*CarNo*
*DriverNo*
*WeekNo*
Total_Kilometers
Total_Fuel_Used
Total_Income

DriverDIM
DriverNo
Name
LicenseNum
ExpiryDate

# Summary

# Summary

- A Star Schema with Multi-Fact or Multiple Star Schemas are needed because each Fact focuses on a subject or on a different granularity.
- Star schemas with different dimensions can be said to have different subjects.
- When multiple star schemas have different subjects (or different dimensions), they cannot be merged into one star schema.

# Lecture Activity

MONASH
University

# Data Cleaning

# Pre-Data Warehousing: Exploring Dirty Data

**Data cleaning** is to clean data that is dirty. The crucial step in data cleaning is to identify data incorrectness, inconsistency, etc. before loading the data into the data warehouse

**For example:**

1. Duplication problems
2. Relationship problems
3. Inconsistent values
4. Incorrect values
5. The *null value* problems

# 1. Duplicate Problem

- **To check duplicate records**

```
SELECT <<PK attribute>>, COUNT(*)
FROM <<operational database table>>
GROUP BY <<PK attribute>>
HAVING COUNT(*) > 1;
```

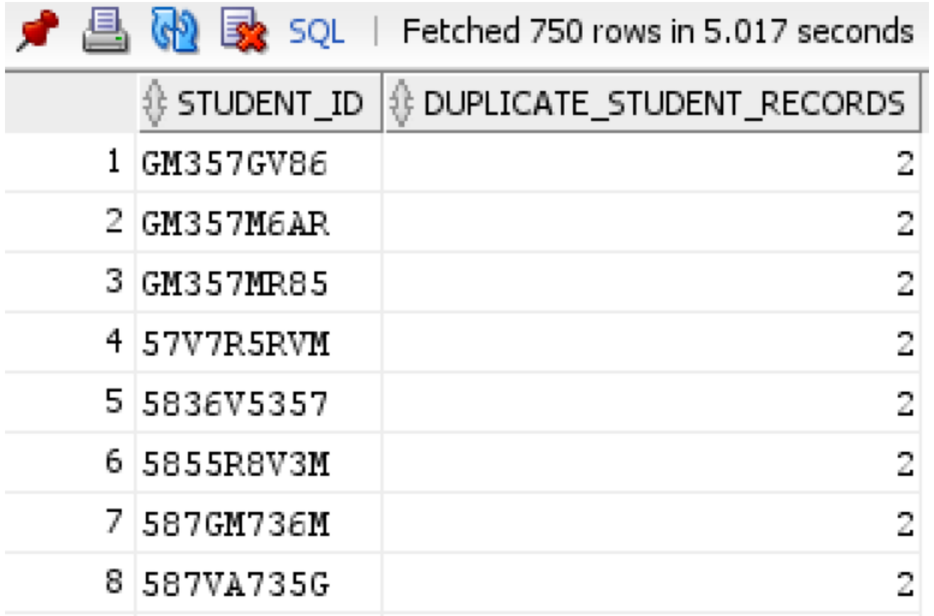
- **To clear duplicate records**

```
CREATE TABLE <<new table>> AS
SELECT DISTINCT *
              FROM <<old table>>;
```

# 1. Duplicate Problem - UseLog case study

### To check duplicate records in Student table

```
SELECT student_id, COUNT(*) as duplicate_Student_records
FROM dw.student
GROUP BY student_id
HAVING COUNT (*) >1;
```



SQL | Fetched 750 rows in 5.017 seconds

| | STUDENT_ID | DUPLICATE_STUDENT_RECORDS |
|---|---|---|
| 1 | GM357GV86 | 2 |
| 2 | GM357M6AR | 2 |
| 3 | GM357MR85 | 2 |
| 4 | 57V7R5RVM | 2 |
| 5 | 5836V5357 | 2 |
| 6 | 5855R8V3M | 2 |
| 7 | 587GM736M | 2 |
| 8 | 587VA735G | 2 |

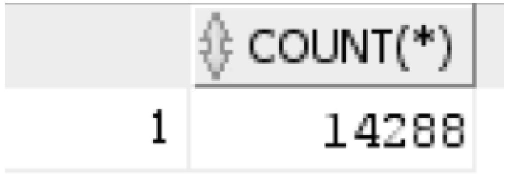
### To clean duplicate records:

```
CREATE TABLE student AS
SELECT DISTINCT *
FROM dw.student;
```

### To check before and after cleaning:

**Before:**

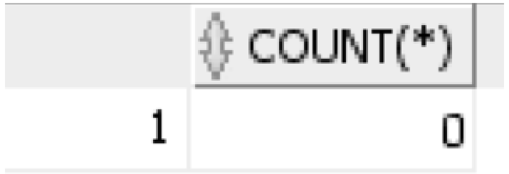
```
SELECT COUNT(*)
FROM (SELECT student_id,
COUNT(*)
    FROM dw.student
    GROUP BY student_id
    HAVING COUNT(*) > 1);
```



| | COUNT(*) |
|---|---|
| 1 | 14288 |

**After:**

```
SELECT COUNT(*)
FROM (SELECT student_id,
COUNT(*)
    FROM student
    GROUP BY student_id
    HAVING COUNT(*) > 1);
```



| | COUNT(*) |
|---|---|
| 1 | 0 |

# 2. Relationship Problem

- **To check invalid FK values**

```
SELECT *
FROM <<table 1>>
WHERE <<FK>> NOT IN
            (SELECT <<PK>>
            FROM <<table 2>>);
```

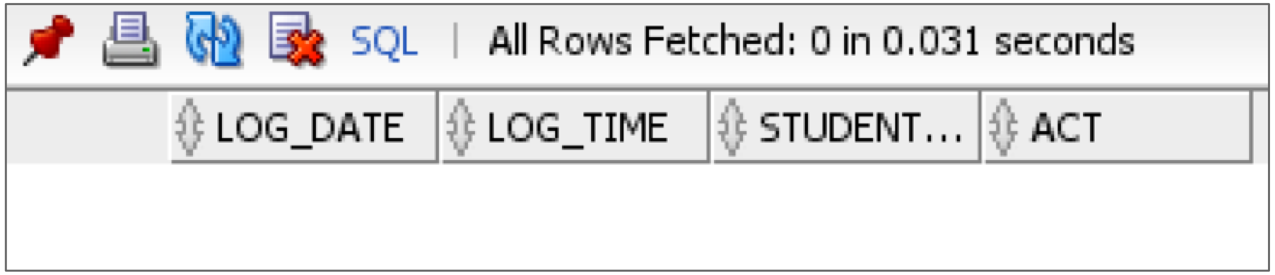
- **To resolve this issue (simplest approach)**

```
DELETE
FROM <<table 1>>
WHERE <<FK>> NOT IN
            (SELECT <<PK>>
            FROM <<table 2>>);
```

# 2. Relationship Problem - UseLog case study

**To check if there are any illegal students in Uselog table who are not in Student table**

```
SELECT *
FROM dw.uselog
WHERE student_id NOT IN
        ( SELECT student_id
        FROM dw.student);
```



All Rows Fetched: 0 in 0.031 seconds

| LOG_DATE | LOG_TIME | STUDENT... | ACT |
|---|---|---|---|

**If there is any illegal record, to resolve the issue (simplest approach):**

```
CREATE TABLE uselog AS
SELECT *
FROM dw.uselog;

CREATE TABLE student AS
SELECT *
FROM dw.student;
```

```
DELETE
FROM uselog
WHERE student_id NOT IN
                (SELECT student_id
                FROM student);
```

# 3. Inconsistent Values

Inconsistent values can be different in units, codes or different precisions such as two different attributes conflict to each other

<u>**For example:**</u> Cable Television case study - Workshop activity week 4
Error: Contract Start Time > Contract End Time

**To check if there is any inconsistent values:**

```
SELECT *
FROM contract
WHERE starttime > endtime
```

**If there is any inconsistent values, to resolve the issue (simplest approach):**

```
DELETE
FROM contract
WHERE starttime > endtime;
```

# 4. Incorrect Values

Incorrect values can be incorrect spelling, attributes fall outside the correct data range, illogical value of the attribute or incorrect data type

**For example:** Robcor case study

**To check if there is any incorrect values:**

```
SELECT *
FROM charter
WHERE char_distance < 0;
```

**If there is any inconsistent values, to resolve the issue (simplest approach):**

```
DELETE
FROM charter
WHERE char_distance <0;
```

# 5. Null Value Problems

**<u>For example:</u>** Uselog case study

**To check if there is any null value in Major table:**

```
SELECT *
FROM major
WHERE major_name IS NULL;
```

**If there is any null value, to resolve the issue (simplest approach):**

```
DELETE
FROM major
WHERE major_name IS NULL;
```