FIT3003
Major Assignment

SEMESTER 2 2022 (WEIGHT = 20%)

COMPLETED BY

Jason Ching Yuen, SIU (31084222)
Patty Pei Chi, HUNG (29302951)

Contribution Declaration Form
(to be completed by all team members)

Please fill in the form with the contribution from each student towards the assignment.

1 NAME AND CONTRIBUTION DETAILS

Student ID	Student Name	Contribution Percentage
31084222	Jason Ching Yuen Siu	50%
29302951	Pei Chi Hung (Patty)	50%

2 DECLARATION

We declare that:

- The information we have supplied in or with this form is complete and correct.
- We understand that the information we have provided in this form will be used for individual assessment of the assignment.

3 SIGNATURE

Signatures

Jason Ching Yuen Siu

Pei-Chi Hung

Date

Day Month Year

09 / 10 /2022

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
31084222	Siu	**Jason Ching Yuen**
29302951	Hung	**Pei-Chi**

***** Please include the names of all other group members.

Unit name and code	***FIT3003 - Business intelligence and data warehousing***	
Title of assignment	FIT3003 Major Assignment	
Lecturer/tutor	Bao and Arif	
Tutorial day and time	Thursday, 6-8 pm	**Campus** Clayton
Is this an authorised group assignment?	☐ **Yes** ☐ **No**	
Has any part of this assignment been previously submitted as part of another unit/course?	☐ **Yes** ☐ **No**	
Due Date 12/10/22		**Date submitted Before 12/10/22**

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date) ..... Signature of lecturer/tutor .....

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - i. provide to another member of faculty and any external marker; and/or
  - ii. submit it to a text matching software; and/or
  - iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature ..... *Date*……

* delete (iii) if not applicable

Signature ____Jason Ching Yuen Siu _____ Date:_____9/10/2022_____

Signature ____Pei-Chi Hung_____ Date:_____ 9/10/2022_____

Signature _____Date:_____ Signature _____Date:_____

Signature _____Date:_____ Signature _____Date:_____

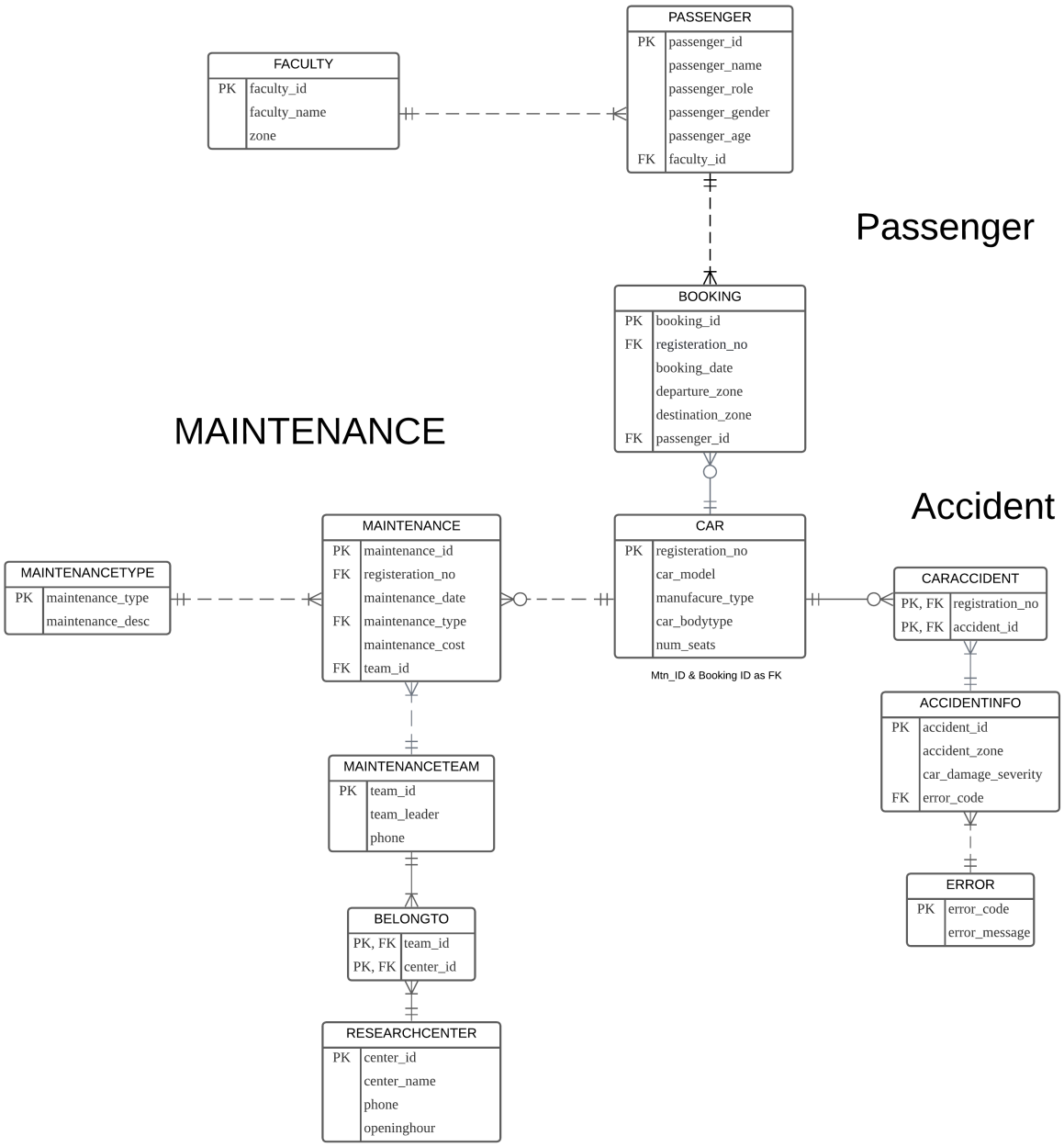
MONASH University
Information Technology

Privacy Statement
The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

Item	Jason Ching Yuen Siu	Hung Pei Chi
C1 Data Warehouse Design and Transformation		
ER Diagram	Together	Together
Data Cleaning	3 problems / 5 problems	2 problems / 5 problems
Star Schema Version-1 with explanation on difference	Wrote this	
Star Schema Version-2 with explanation on difference		Wrote this
Determinant dimension explanation	Wrote this	
Temporal dimension explanation		Wrote this
C2 Star/Snowflake Implementation		
Star Schema Version-1 implementation	Together	Together
Star Schema Version-2 implementation	Together	Together
C3 Reports using OLAP Queries		
OLAP Reports (Reports 1 to 4)	Together, in charge of report 1 and 2	Together, in charge of report 3 and 4
Report with Roll up and Partial Roll up (Reports 5 to 6)	Wrote this	
Report with Moving and cumulative aggregates (Reports 7 to 8)		Wrote this
C4 BI Reports&dashboard		
A dashboard with report 2 and 4	Completed this	
A dashboard with report 7 and 8		Completed this
C5 Final Recommendations/Suggestions		
Final suggestion	Make interpretation	Summarise the data statistics
MISC		
Report formatting	Together, in charge of this	Together, in charge of this
Code double-checking	Together	Together, in charge of this

1.1. The E/R diagram of the operational database

The following is the ER diagram of the required case.



1.2. Data cleaning process

Cleaning strategies we used

Before operational databases are ready to be transformed to a data warehouse, we need to identify dirty data and clean them — a crucial step in the activities of pre-data warehousing.

We based the week 4's contents where we learnt five aspects of problems and run those queries we learnt one by one through every entity in the operational database. The reason why we do this — despite not being efficient — is that this is an effective approach where we covered all aspects of problems and ensure that all entitles are CLEANED.

Full SQL script for cleaning data

```
-- ============================  Problem 1: Duplicate
SELECT BOOKINGID, COUNT(*) as duplicate_records
FROM MonCity.BOOKING
GROUP BY BOOKINGID
HAVING COUNT (*) >1;

-- solution
DROP TABLE Clean_Booking CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_Booking as
SELECT DISTINCT *
FROM MonCity.BOOKING;


-- ============================  Problem 2: Null value problem
SELECT *
FROM moncity.ACCIDENTINFO
WHERE ACCIDENTID IS NULL;

-- solution
DROP TABLE Clean_Accidentinfo CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_Accidentinfo as
SELECT *
FROM MonCity.ACCIDENTINFO
WHERE ACCIDENTID IS NOT NULL;

-- ============================  Problem 3: Incorrect Values
SELECT *
FROM moncity.maintenance
WHERE maintenancecost < 0;

-- solution
DROP TABLE Clean_maintenance CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_maintenance as
```

```
SELECT *
FROM MonCity.MAINTENANCE
WHERE MAINTENANCECOST > 0;

-- ============================  Problem 4: Relationship Problem
SELECT ERRORcode
FROM ACCIDENTINFO
WHERE ERRORcode  NOT IN ( SELECT ERRORcode FROM MonCity.ERROR );

-- solution
DELETE
FROM CLEAN_ACCIDENTINFO
WHERE ERRORcode NOT IN ( SELECT ERRORcode FROM MonCity.ERROR );

-- ============================  Problem 5: Relationship Problem
SELECT *
FROM MONCITY.passenger
WHERE FACULTYID NOT IN
(SELECT FACULTYID
FROM moncity.faculty);

-- solution
DROP TABLE Clean_passenger CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_passenger as
SELECT DISTINCT *
FROM MonCity.passenger;

DELETE
FROM Clean_passenger
WHERE FACULTYID NOT IN
( SELECT FACULTYID
FROM moncity.faculty );
```

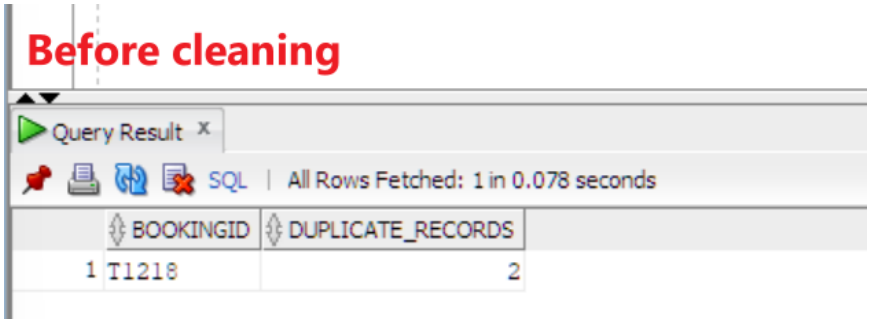
Demonstration of the difference between before and after cleaning

At the end, 5 types of problem are identified corresponding to the problems illustrated below.

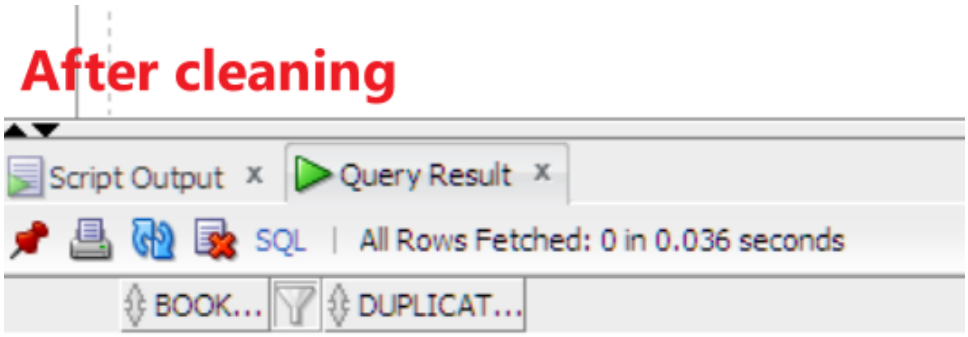
- Problem 1: Duplicate
- Problem 2: Null value problem
- Problem 3: Incorrect Values
- Problem 4: Relationship Problem and Inconsistent Values
- Problem 5: Relationship Problem

Here we demonstrated 5 specific problems we found:

▼ Problem 1: Two duplicate entries found in `MonCity.BOOKING` where `bookingid = 'T1218'`

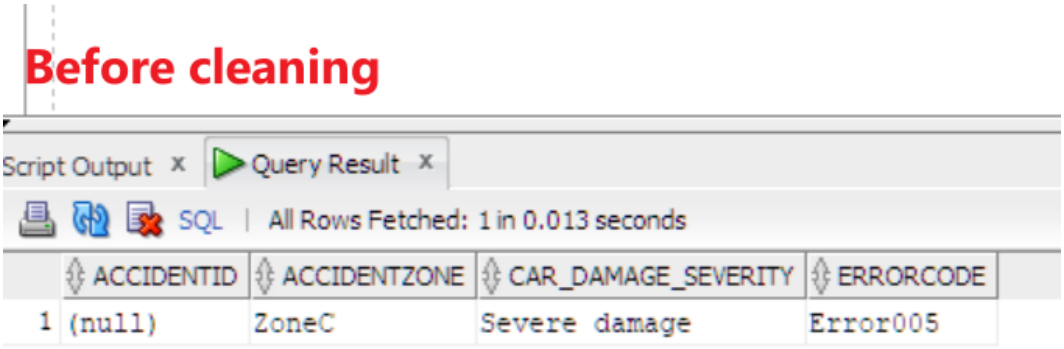



```
SELECT BOOKINGID, COUNT(*) as
duplicate_records
FROM MonCity.BOOKING
GROUP BY BOOKINGID
HAVING COUNT (*) >1;
```

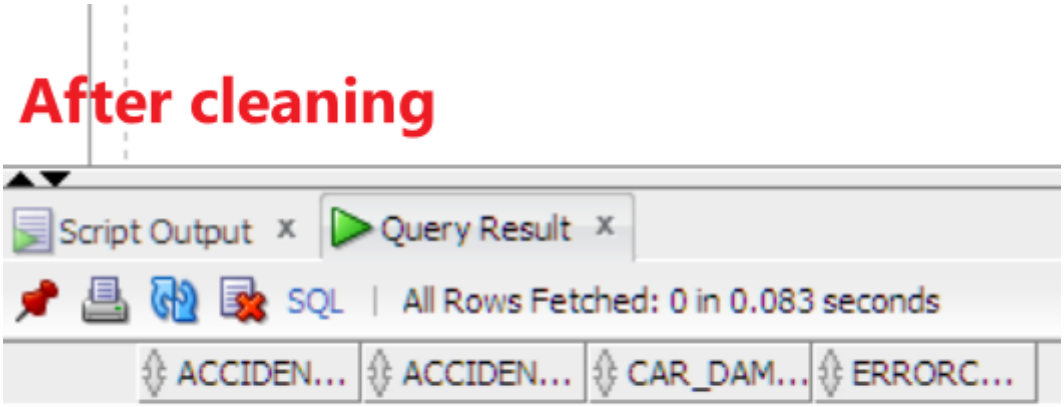



```
DROP TABLE Clean_Booking CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_Booking as
SELECT DISTINCT *
FROM MonCity.BOOKING;

SELECT BOOKINGID,
COUNT(*) as duplicate_records
FROM Clean_Booking
GROUP BY BOOKINGID
HAVING COUNT (*) >1;
```

▼ Problem 2: One entry found with NULL value



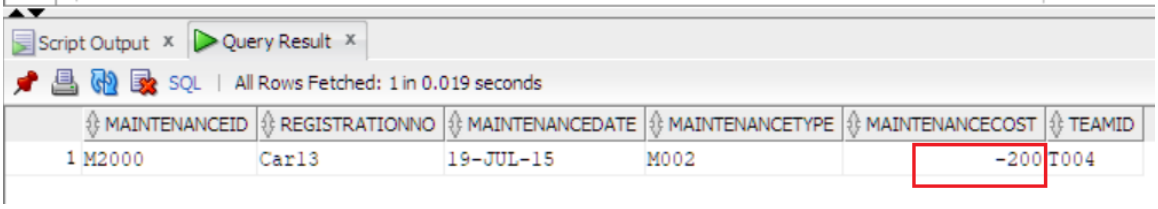

```
SELECT *
FROM moncity.ACCIDENTINFO
WHERE ACCIDENTID IS NULL;
```




```
DROP TABLE Clean_Accidentinfo CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_Accidentinfo as
SELECT *
FROM MonCity.ACCIDENTINFO
WHERE ACCIDENTID IS NOT NULL;

SELECT *
FROM Clean_Accidentinfo
WHERE ACCIDENTID IS NULL;
```

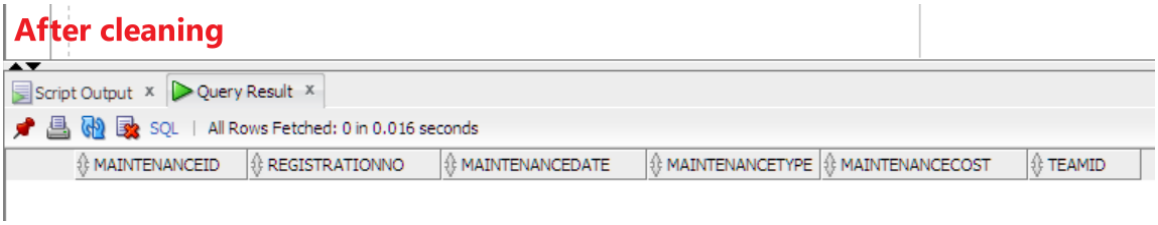
▼ Problem 3: One entry found with negative amount of maintenance cost where `maintenanceid = 'M2000'`

Before cleaning




```
SELECT *
FROM moncity.maintenance
WHERE maintenancecost < 0;
```

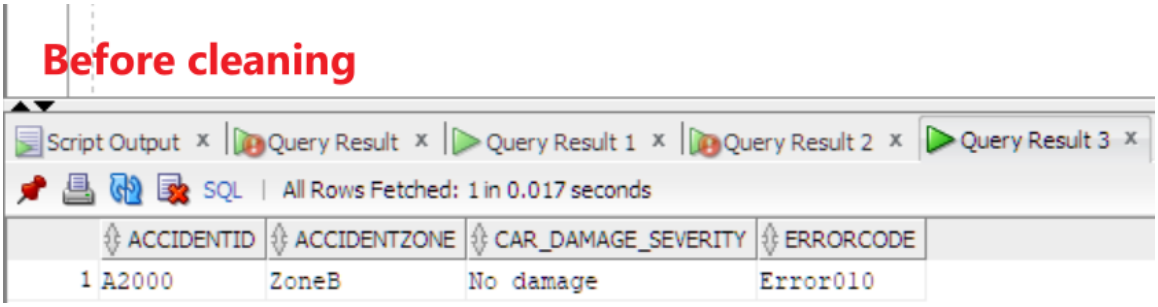
After cleaning




```
DROP TABLE Clean_maintenance CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_maintenance as
SELECT *
FROM MonCity.MAINTENANCE
WHERE MAINTENANCECOST > 0;

SELECT *
FROM Clean_maintenance
WHERE maintenancecost < 0;
```

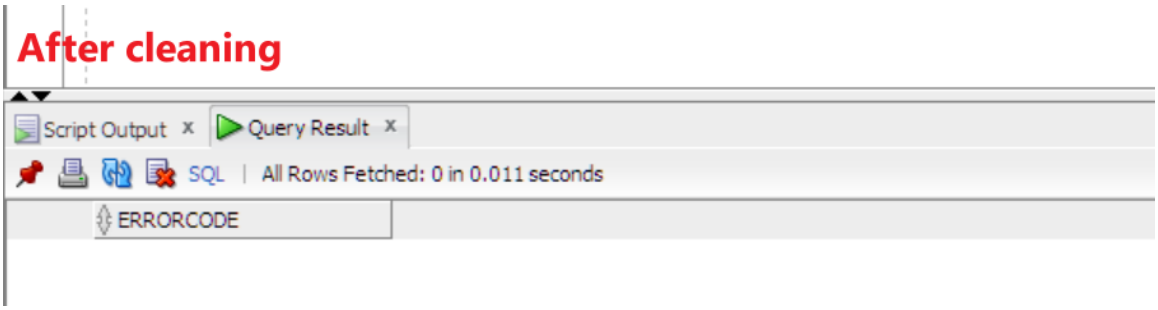
▼ Problem 4: One entry found `MONCITY.ACCIDENTINFO` in where `accidentid = 'A2000'` does not exist in `MONCITY.error` AND `error code = 'Error010'` which is inconsistent

Before cleaning



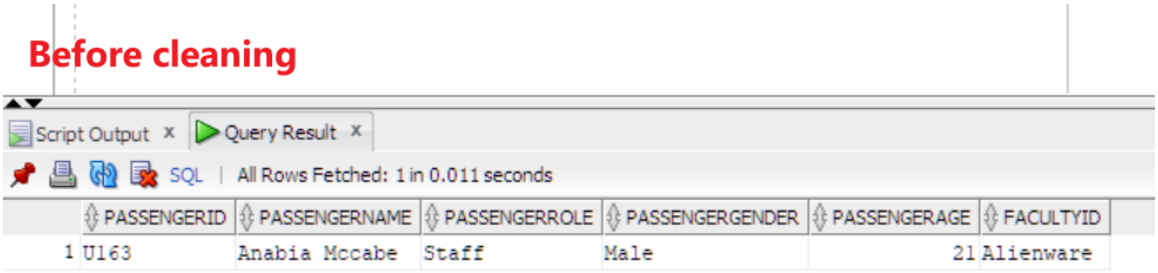

```
SELECT ERRORcode
FROM ACCIDENTINFO
```

```
WHERE ERRORcode  NOT IN ( SELECT ERRORcode FROM MonCity.ERROR );
```

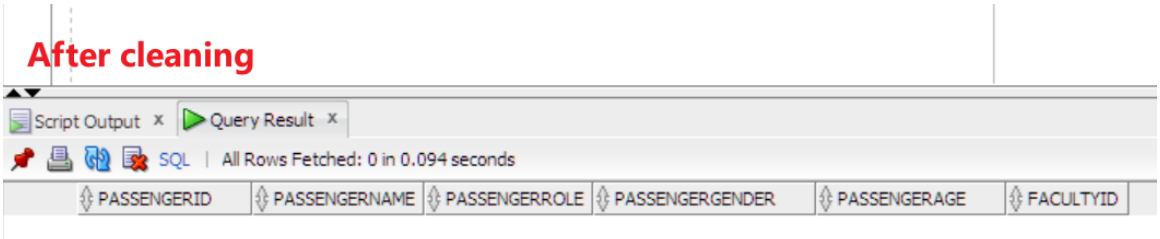



```
DELETE
FROM CLEAN_ACCIDENTINFO
WHERE ERRORcode NOT IN ( SELECT ERRORcode FROM MonCity.ERROR );

SELECT ERRORcode
FROM CLEAN_ACCIDENTINFO
WHERE ERRORcode  NOT IN ( SELECT ERRORcode FROM MonCity.ERROR );
```

▼ Problem 5: One entry found `MONCITY.passenger` in where `passenger_id = 'U163'` does not exist in `MONCITY.faculty`



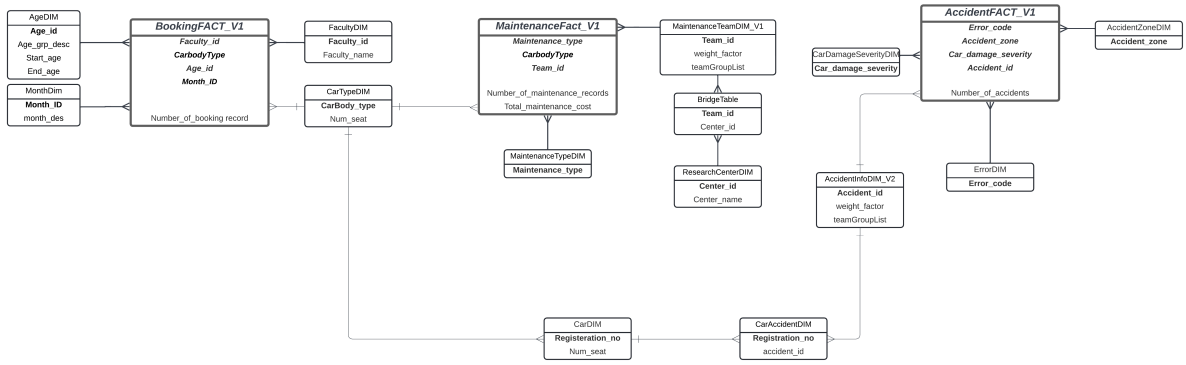

```
SELECT *
FROM MONCITY.passenger
WHERE FACULTYID NOT IN
(SELECT FACULTYID
FROM moncity.faculty);
```



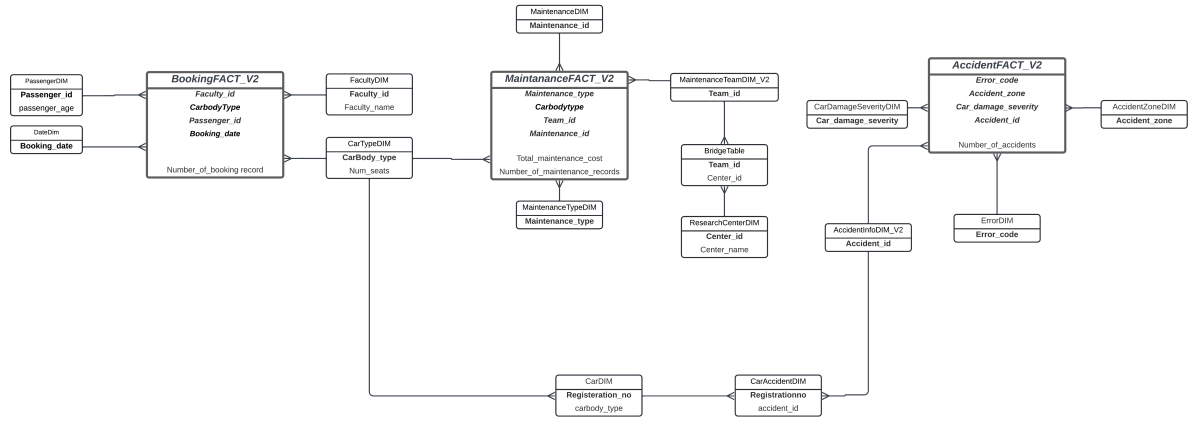
```
DROP TABLE Clean_passenger CASCADE CONSTRAINTS PURGE;
CREATE TABLE Clean_passenger as
SELECT DISTINCT *
FROM MonCity.passenger;

DELETE
FROM Clean_passenger
WHERE FACULTYID NOT IN
( SELECT FACULTYID
FROM moncity.faculty );

SELECT *
FROM Clean_passenger WHERE FACULTYID NOT IN
(SELECT FACULTYID
FROM moncity.faculty);
```

1.3. Star schema designs

1.3.1. Star schema design version 1 (Level 1)



1.3.1. Star schema design version 2 (Level 0)



1.4 Exclusion determinant dimension and temporal dimension

The outputs of this task are:

c) Two versions of star/snowflake schema diagrams.

d) 1. The reasons for the choice of determinant dimension(s) in your star schema, or the reason for its absence.

2. The reasons for the choice of SCD type(s) for any temporal dimensions in your star schema, or the reason for its absence.

e) An explanation of the differences between the two versions of star/snowflake schemas.

Note: *The above explanation must be consistent with your star schema diagram and based on the assignment scenario. Please have a maximum of 300 words for each explanation.*

The section justifies the reasons why we did not use determinant dimension and temporal dimension.

Absence of determinant dimension

A determinant dimension is one on which the fact measure relies on, so all data retrieval from DWs must include this dimension, lest the retrieval data is not reasonable. There is no specific key ID from a dimension that the fact measure needs to rely on, for one reason:

Because of the on the requirement of report from manager, we used the aggregated functions like `COUNT` or `SUM`, and the information of these functions does **NOT** rely on a single dimension. That is, without a specific dependence on a single dimension, we can still make reasonable data retrieval and analyse the fact measures (e.g., total count of booking records). Therefore, we did **NOT** include a determinant dimension.

Absence of temporal dimension

Incorporating temporal dimension allows the DWs to have a temporal aspect for records. It is a mechanisms for managing information that varies over time. Base on the scenario, attributes included dates are the dates of maintenance, accident, and booking. And the attributes attached with these dates will not evolve over time. For example, attributes like the cost of a specific maintenance record, number of seats for a specific car, and the car severity of a accident record do not change over time normally. Since the transaction time is its own single lifespan, it is reasonable to assume that dimensions are stable, i.e., their data do not change. And because of this, we do **NOT** include a temporal dimension to keep track of the evolution of dimensions, facts, and measures.

The difference between the two versions of star schema

- lower agedim to passenger dim
- lower monthdim to datedim
- add in maintenance dim

2.1. Star schema implementation - Level 1

The following is the SQL script for implementing the star schema of version 1 (Level 1).

```
-- ========================= SharedDim ==============================

-- DIM: CarbodyDIM
DROP TABLE CarBodyTypeDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarBodyTypeDIM AS
(SELECT DISTINCT(CARBODYTYPE), numseats FROM moncity.car);

-- DIM: CarDim
DROP TABLE CarDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarDIM AS
(SELECT DISTINCT registrationno, CARBODYTYPE, numseats
FROM moncity.car);

-- ========================= AccidentFact ==============================

-- DIM: AccidentZoneDIM
DROP TABLE AccidentZoneDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE AccidentZoneDIM AS
(SELECT DISTINCT accidentzone
FROM clean_accidentinfo);

-- DIM: AccidentInfoDIM_V1
DROP TABLE AccidentInfoDIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE AccidentInfoDIM_V1 AS
(
SELECT ai.accidentid,
1.0/count(ca.accidentid) As WeightFactor,
LISTAGG (ca.registrationno, '_') Within Group (Order By ai.accidentid) As TeamGroupLis
t
FROM clean_accidentinfo ai, moncity.caraccident ca
WHERE ai.accidentid = ca.accidentid
Group By ai.accidentid
);

-- DIM: CarAccidentDIM
DROP TABLE CarAccidentDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarAccidentDIM AS
(SELECT REGISTRATIONNO, accidentid FROM moncity.caraccident);

-- DIM: ErrorDIM
DROP TABLE ErrorDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE ErrorDIM AS
(SELECT errorcode FROM moncity.error);

-- DIM: CarDamageSeverityDIM
```

```
DROP TABLE CarDamageSeverityDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarDamageSeverityDIM AS
(SELECT DISTINCT (CAR_DAMAGE_SEVERITY)
FROM clean_accidentinfo);



-- ======================== MaintenanceFact ==============================

-- DIM: MaintenanceTypeDIM
DROP TABLE MaintenanceTypeDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE MaintenanceTypeDIM AS
(SELECT maintenancetype
FROM moncity.maintenancetype);

-- DIM: (B) MaintenanceTeamDIM_V1
DROP TABLE MaintenanceTeamDIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MaintenanceTeamDIM_V1 AS
(
SELECT T.TEAMID,
1.0/count(B.CENTERID) As WeightFactor,
LISTAGG (B.CENTERID, '_') Within Group (Order By B.CENTERID) As TeamGroupList
FROM moncity.maintenanceTeam T, moncity.belongto B
WHERE T.TEAMID = B.TEAMID
Group By T.TEAMID
);

-- DIM: (B) BridgeTable
DROP TABLE BridgeTable CASCADE CONSTRAINTS PURGE;
CREATE TABLE BridgeTable
AS (SELECT * FROM moncity.belongto);

-- DIM: (B) ResearchCenterDIM
DROP TABLE ResearchCenterDIM  CASCADE CONSTRAINTS PURGE;
CREATE TABLE ResearchCenterDIM
AS(
SELECT CENTERID, CENTERNAME
FROM moncity.researchcenter
);

-- ======================== BookingFact ==============================

-- DIM: FacultyDIM
DROP TABLE FacultyDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE FacultyDIM AS
(SELECT FACULTYID, FACULTYNAME
FROM moncity.faculty);

-- DIM: (M) MonthDIM
DROP TABLE MonthDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE MonthDIM AS
SELECT distinct to_char(BOOKINGDATE, 'MM') as MonthID,
to_char(BOOKINGDATE, 'MONTH') as Month_Des
FROM Clean_Booking;

-- DIM: (M) AgeDim
DROP TABLE AgeDim CASCADE CONSTRAINTS PURGE;
CREATE TABLE AgeDim
```

```
(AgeID varchar2(10),
Age_grp_desc varchar2(50),
Start_age  number(3),
End_age number(3));

-- Insert age group

Insert into AgeDim values ('Group 1', 'Young adult', 18, 35);
Insert into AgeDim values ('Group 2', 'Middle adult', 36, 59);
Insert into AgeDim values ('Group 3', 'Old-aged adult', 60, 110);



--------------------------------------------------------------------------------
-- Creating facts:
--------------------------------------------------------------------------------

-- ========================  FACT: AccidentFACT  ========================

DROP TABLE AccidentFACT_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE AccidentFACT_V1
AS (
SELECT accidentzone,
errorcode,
car_damage_severity,
accidentid,
count(accidentid) as num_of_accident
FROM clean_accidentinfo
GROUP BY
accidentzone,
car_damage_severity,
errorcode,
accidentid
);


-- ========================  FACT: BookingFact  ========================
-- 1. Create BookingTempFact
DROP TABLE BookingTempFact CASCADE CONSTRAINTS PURGE;
    CREATE TABLE BookingTempFact
AS (
SELECT to_char(bookingdate, 'MM') as MonthID, f.facultyid, c.carbodytype, p.passengera
ge, b.bookingid
FROM clean_booking b, clean_passenger p, moncity.faculty f, moncity.car c
WHERE b.passengerid  = p.passengerid AND f.facultyid = p.facultyid AND b.REGISTRATIONN
O = c.REGISTRATIONNO
);

-- 2. Create columns of AgeID and TimeID

ALTER table BookingTempFact
ADD(
AgeID varchar(15)
);


-- 3. Update values of ageDim
Update BookingTempFact
```

```
Set ageid = 'Group 1'
WHERE ( passengerage between 18 AND 35);

Update BookingTempFact
Set ageid = 'Group 2'
WHERE ( passengerage between 36 AND 59);

Update BookingTempFact
Set ageid = 'Group 3'
WHERE ( passengerage between 60 AND 110);



-- 4. Create BookingFACT
DROP TABLE BookingFACT_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE BookingFACT_V1
AS (
SELECT b.FACULTYID, b.CARBODYTYPE, b.AGEID , b.MONTHID,
count (b.BOOKINGID) as num_of_booking
FROM bookingtempfact b
GROUP BY b.FACULTYID, b.CARBODYTYPE, b.AGEID , b.MONTHID
);


-- ========================  FACT: MaintenanceFact  ========================
DROP TABLE MaintenanceFact_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MaintenanceFact_V1
AS (
SELECT m.maintenancetype,
carbodytype,m.teamid,

count (DISTINCT  m.maintenanceid) as num_of_main_record,
sum(m.maintenancecost) as main_cost

FROM clean_maintenance m,

(SELECT DISTINCT  b.teamid FROM
moncity.maintenanceteam mte, moncity.belongto b, moncity.researchcenter r )  mTeam,

moncity.maintenancetype mty,
moncity.car c

WHERE m.teamid = mTeam.teamid AND
mty.maintenancetype = m.maintenancetype AND
c.registrationno = m.registrationno

GROUP BY m.maintenancetype, carbodytype,m.teamid
);
```

2.2. Star schema implementation - Level 0

The following is the SQL script for implementing the star schema of version 2 (Level 0).

```
-- ======================== ShareDIM ================================
-- DIM: CarbodyDIM
DROP TABLE CarBodyTypeDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarBodyTypeDIM AS
(SELECT DISTINCT(CARBODYTYPE), numseats FROM moncity.car);

-- ======================== BookingFact ==============================
-- DIM: PassengerDIM
DROP TABLE PassengerDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE PassengerDIM
  AS (SELECT PASSENGERID, PASSENGERAGE
FROM Clean_passenger
);

-- DIM: DateDIM
DROP TABLE DateDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE DateDIM
AS (SELECT BOOKINGDATE
FROM Clean_Booking
);

-- DIM: FacultyDIM
DROP TABLE FacultyDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE FacultyDIM AS
(SELECT FACULTYID, FACULTYNAME
FROM moncity.faculty);

-- ======================== MaintenanceFact ==============================
-- DIM: MaintenanceTypeDIM
DROP TABLE MaintenanceTypeDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE MaintenanceTypeDIM AS
(SELECT maintenancetype
FROM moncity.maintenancetype);

-- DIM: (B) MaintenanceTeamDIM_V2
DROP TABLE MaintenanceTeamDIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MaintenanceTeamDIM_V2 AS
(SELECT TEAMID
FROM moncity.maintenanceTeam
);

-- DIM: (B) BridgeTable
DROP TABLE BridgeTable CASCADE CONSTRAINTS PURGE;
CREATE TABLE BridgeTable
AS (SELECT * FROM moncity.belongto);
```

```
-- DIM: (B) ResearchCenterDIM
DROP TABLE ResearchCenterDIM  CASCADE CONSTRAINTS PURGE;
CREATE TABLE ResearchCenterDIM
AS(
SELECT CENTERID, CENTERNAME
FROM moncity.researchcenter
);

-- DIM: MaintenanceDIM
DROP TABLE MaintenanceDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE MaintenanceDIM
AS(
SELECT maintenanceid
FROM Clean_maintenance
);

-- ======================== AccidentFact ==============================

-- DIM: AccidentZoneDIM
DROP TABLE AccidentZoneDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE AccidentZoneDIM AS
(SELECT DISTINCT accidentzone
FROM clean_accidentinfo);

-- DIM: AccidentInfoDIM_V2
DROP TABLE AccidentInfoDIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE AccidentInfoDIM_V2 AS
(
SELECT clean_accidentinfo.accidentid
FROM clean_accidentinfo
);

-- DIM: CarAccidentDIM
DROP TABLE CarAccidentDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarAccidentDIM AS
(SELECT REGISTRATIONNO, accidentid FROM moncity.caraccident);

-- DIM: ErrorDIM
DROP TABLE ErrorDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE ErrorDIM AS
(SELECT errorcode FROM moncity.error);

-- DIM: CarDamageSeverityDIM
DROP TABLE CarDamageSeverityDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarDamageSeverityDIM AS
(SELECT DISTINCT (CAR_DAMAGE_SEVERITY)
FROM clean_accidentinfo);

-- DIM: CarDIM
DROP TABLE CarDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE CarDIM AS
(SELECT DISTINCT registrationno, CARBODYTYPE
FROM moncity.car);

-------------------------------------------------------------------------------
-- Creating facts:
-------------------------------------------------------------------------------
```

```
-- ========================  FACT: AccidentFACT  ========================
-- ==== new AccidentFACT
DROP TABLE AccidentFACT_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE AccidentFACT_V2
AS (
SELECT accidentzone, errorcode, car_damage_severity, accidentid,
count(accidentid) as num_of_accident
FROM clean_accidentinfo
GROUP BY
accidentzone,
car_damage_severity,
errorcode,
accidentid
);


-- ========================  FACT: BookingFact  ========================
-- Create BookingFact
DROP TABLE BookingFACT_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE BookingFACT_V2
AS (
SELECT b.bookingdate, f.facultyid, c.carbodytype, p.passengerage, b.bookingid, count(b
ookingid) as num_of_booking_record
FROM Clean_Booking b, Clean_passenger p, moncity.faculty f, moncity.car c
WHERE b.passengerid  = p.passengerid AND f.facultyid = p.facultyid AND b.REGISTRATIONN
O = c.REGISTRATIONNO
GROUP BY b.bookingdate, f.facultyid, c.carbodytype, p.passengerage, b.bookingid
);

-- ========================  FACT: MaintenanceFact  ========================
DROP TABLE MaintenanceFact_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MaintenanceFact_V2
AS (
SELECT m.maintenanceid,
m.maintenancetype,
c.carbodytype,
m.teamid,
count(DISTINCT  m.maintenanceid) as num_of_main_record,
sum(m.maintenancecost) as main_cost

FROM Clean_maintenance m,

(SELECT DISTINCT  b.teamid FROM
moncity.maintenanceteam mte, moncity.belongto b, moncity.researchcenter r )  mTeam,

moncity.maintenancetype mty,
moncity.car c

WHERE m.teamid = mTeam.teamid AND
mty.maintenancetype = m.maintenancetype and
c.registrationno = m.registrationno


GROUP BY m.maintenanceid, m.maintenancetype, c.carbodytype,m.teamid
);
```

3.1. OLAP queries

The following is the SQL script for making the report 1, 2, 3 and 4.

REPORT 1

MonCity’s cumulative number of booking records of each month for Faculty of IT

	FACULTYID	MONTH_DES	TOTAL_BOOKING	CUM_BOOKING
1	FIT	JANUARY	260	260
2	FIT	FEBRUARY	230	490
3	FIT	MARCH	234	724
4	FIT	APRIL	228	952
5	FIT	MAY	245	1,197
6	FIT	JUNE	252	1,449
7	FIT	JULY	249	1,698
8	FIT	AUGUST	245	1,943
9	FIT	SEPTEMBER	274	2,217
10	FIT	OCTOBER	256	2,473
11	FIT	NOVEMBER	251	2,724
12	FIT	DECEMBER	251	2,975

```
SELECT b.facultyid, m.Month_Des, SUM(b.num_of_booking) as Total_Booking,
TO_CHAR (SUM(SUM(num_of_booking)) OVER
(ORDER BY b.facultyid, to_date(m.Month_Des, 'Month')
ROWS UNBOUNDED PRECEDING),
'9,999,999,999') AS CUM_booking
FROM bookingfact_V1 b, monthdim m
WHERE b.facultyid = 'FIT' and b.monthid = m.monthid
GROUP BY b.facultyid, m.Month_Des
ORDER BY MIN(m.monthid);
```

REPORT 2

MonCity’s maintenance report

	TEAMID	CARBODYTYPE	TOTAL_NUMBER_OF_MAINTENANCE	TOTAL_MAINTENANCE_COST
1	All Team	All Car Body Types	399	125300
2	All Team	Bus	136	44900
3	All Team	Mini Bus	113	34000
4	All Team	People Mover	150	46400
5	T002	All Car Body Types	197	62700
6	T002	Bus	58	18400
7	T002	Mini Bus	62	19300
8	T002	People Mover	77	25000
9	T003	All Car Body Types	202	62600
10	T003	Bus	78	26500
11	T003	Mini Bus	51	14700
12	T003	People Mover	73	21400

```
SELECT
DECODE (GROUPING(teamid), 1, 'All Team', teamid) as teamid,
DECODE (GROUPING(carbodytype), 1, 'All Car Body Types', carbodytype) as carbodytype,
sum(num_of_main_record) as total_number_of_maintenance,
sum(main_cost) as total_maintenance_cost
FROM MaintenanceFact_V1
WHERE TEAMID = 'T002' OR TEAMID ='T003'
GROUP BY CUBE(teamid, carbodytype);
```

REPORT 3

MonCity’s rank analysis for the number of accidents

	ERRORCODE	REGISTRATIONNO	CARBODYTYPE	TOTAL_NUMBER_OF_ACCIDENTS	ACCIDENT_RANK
1	Error001	Car01	Bus	13	1
2	Error001	Car04	Bus	12	2
3	Error001	Car12	Mini Bus	12	2
4	Error001	Car19	Mini Bus	12	2
5	Error001	Car08	Bus	11	3
6	Error001	Car20	Mini Bus	11	3
7	Error002	Car22	People Mover	45	1
8	Error002	Car27	People Mover	42	2
9	Error002	Car30	People Mover	39	3
10	Error002	Car23	People Mover	39	3
11	Error003	Car14	Mini Bus	12	1
12	Error003	Car06	Bus	12	1
13	Error003	Car01	Bus	11	2
14	Error003	Car10	Bus	11	2
15	Error003	Car12	Mini Bus	10	3
16	Error003	Car09	Bus	10	3
17	Error004	Car12	Mini Bus	13	1

```
WITH  report_three as(
Select af.ERRORCODE, bridge.REGISTRATIONNO,
bridge.CARBODYTYPE, sum(af.NUM_OF_ACCIDENT) AS Total_number_of_accidents,
    DENSE_RANK() OVER (
    PARTITION BY ERRORCODE
```

```
    ORDER BY sum(af.NUM_OF_ACCIDENT) DESC
) AS ACCIDENT_RANK
FROM accidentfact_v1 af,
(
select ca.registrationno as registrationno , carbodytype , ca.accidentid as accidentid
from cardim c , CarAccidentDIM ca, accidentinfodim_V1 ai
where c.registrationno = ca.registrationno AND ai.accidentid = ca.accidentid
) bridge
where af.accidentid  = bridge.accidentid
GROUP BY ERRORCODE, REGISTRATIONNO, CARBODYTYPE
) SELECT *
FROM report_three
WHERE accident_rank in (1,2,3);
```

REPORT 4

MonCity’s booking report

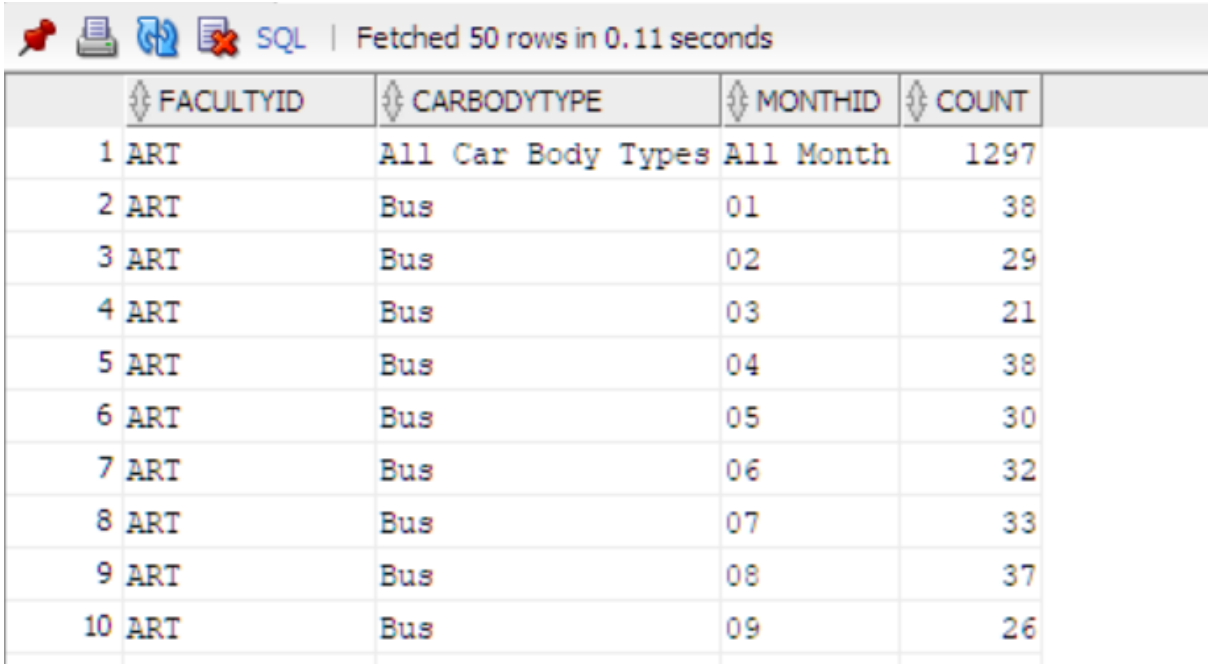
	CARBODYTYPE	AGE_GROUP	FACULTY_ID	TOTAL_BOOKING
1	People Mover	All Age groups	All faculties	3396
2	People Mover	All Age groups	ART	453
3	People Mover	All Age groups	BUS	314
4	People Mover	All Age groups	ENG	841
5	People Mover	All Age groups	FIT	1009
6	People Mover	All Age groups	SCI	779
7	People Mover	G1	All faculties	1380
8	People Mover	G1	ART	169
9	People Mover	G1	BUS	121
10	People Mover	G1	ENG	382
11	People Mover	G1	FIT	390
12	People Mover	G1	SCI	318
13	People Mover	G2	All faculties	1722
14	People Mover	G2	ART	284
15	People Mover	G2	BUS	193
16	People Mover	G2	ENG	429
17	People Mover	G2	FIT	497

```
SELECT carbodytype,
DECODE (GROUPING(ageid), 1, 'All Age groups', ageid) as age_group,
DECODE (GROUPING(facultyid), 1, 'All faculties', facultyid) as faculty_id,
SUM(num_of_booking) as total_booking
FROM BookingFACT_V1 bf
WHERE carbodytype = 'People Mover'
GROUP BY carbodytype, CUBE(ageid, facultyid);
```

3.2. Reports with rollup and partial rollup

The following is the SQL script for making two reports that contain subtotals and one fact measure, using rollup and partial rollup.

REPORT 5: Produce one booking-related report that is useful for management that uses rollup.

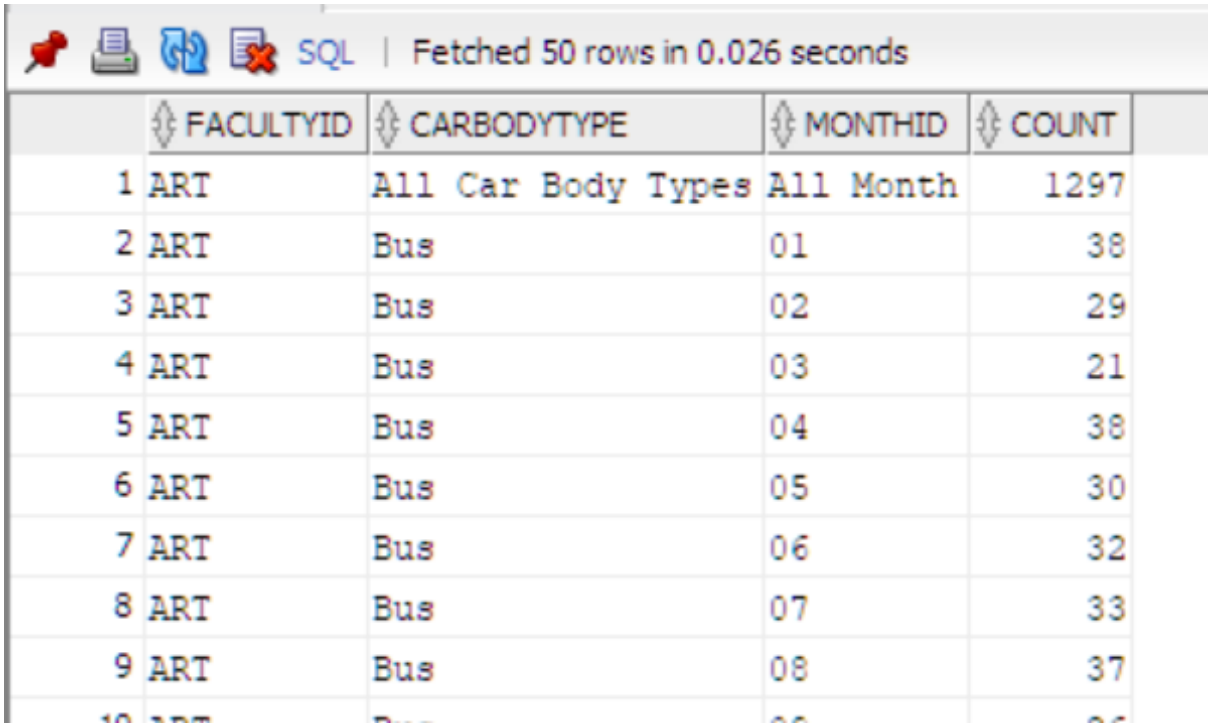


SQL | Fetched 50 rows in 0.11 seconds

	FACULTYID	CARBODYTYPE	MONTHID	COUNT
1	ART	All Car Body Types	All Month	1297
2	ART	Bus	01	38
3	ART	Bus	02	29
4	ART	Bus	03	21
5	ART	Bus	04	38
6	ART	Bus	05	30
7	ART	Bus	06	32
8	ART	Bus	07	33
9	ART	Bus	08	37
10	ART	Bus	09	26

```
-- Report 5
SELECT
DISTINCT DECODE (GROUPING(facultyid), 1, 'All Faculties', facultyid) as facultyid,
DECODE (GROUPING(carbodytype), 1, 'All Car Body Types', carbodytype) as carbodytype,
DECODE (GROUPING(monthid), 1, 'All Month', monthid) as monthid,
SUM(num_of_booking) as count
FROM bookingfact_V1
GROUP BY ROLLUP(facultyid,
carbodytype,
monthid)
ORDER BY facultyid,
carbodytype,
monthid;
```

REPORT 6: Produce one booking-related report that is useful for management that uses partial rollup.



SQL | Fetched 50 rows in 0.026 seconds

	FACULTYID	CARBODYTYPE	MONTHID	COUNT
1	ART	All Car Body Types	All Month	1297
2	ART	Bus	01	38
3	ART	Bus	02	29
4	ART	Bus	03	21
5	ART	Bus	04	38
6	ART	Bus	05	30
7	ART	Bus	06	32
8	ART	Bus	07	33
9	ART	Bus	08	37

```
--Report 6
SELECT
DISTINCT facultyid,
DECODE (GROUPING(carbodytype), 1, 'All Car Body Types', carbodytype) as carbodytype,
DECODE (GROUPING(monthid), 1, 'All Month', monthid) as monthid,
sum(num_of_booking) as count
FROM bookingfact_V1
GROUP BY facultyid, ROLLUP(
carbodytype,
monthid)
ORDER BY facultyid,
carbodytype,
monthid;
```

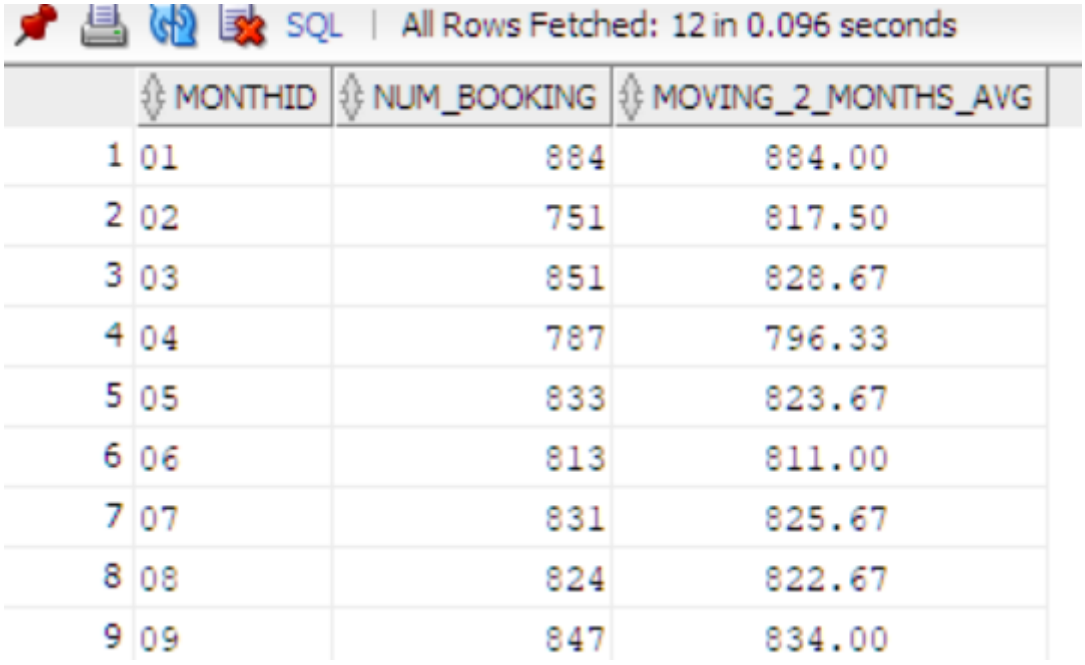
An explanation of the differences between rollup and partial rollup

The difference is that — unlike full rollup — the result of partial rollup does not contain grand total of the booking records.

3.3. Report with moving and cumulative aggregates

The following is the SQL script for making two reports containing moving and cumulative aggregates and one fact measure.

REPORT 7: Produce one moving aggregate report that relates to the Booking information.



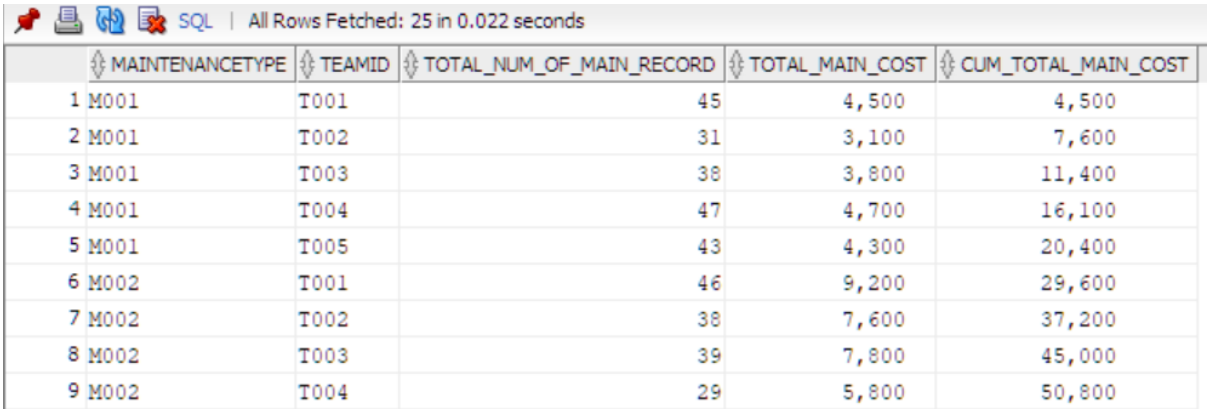
SQL | All Rows Fetched: 12 in 0.096 seconds

	MONTHID	NUM_BOOKING	MOVING_2_MONTHS_AVG
1	01	884	884.00
2	02	751	817.50
3	03	851	828.67
4	04	787	796.33
5	05	833	823.67
6	06	813	811.00
7	07	831	825.67
8	08	824	822.67
9	09	847	834.00

```
Select MONTHID, SUM(NUM_OF_BOOKING) AS Num_Booking,
 TO_CHAR(AVG(SUM(NUM_OF_BOOKING))
 OVER(ORDER BY MONTHID ROWS 2 PRECEDING),
 '9,999,999.99') AS Moving_2_Months_Avg
From bookingfact_V1
Group By MONTHID;
```

This query is valuable because we can use moving average (MA) to smooth out the fluctuation of the trend, and then there are two usages: 1) we can know the which seasons or months are the most and least booked; 2) Using MA to produce a forecasting model, so that we know what is the booking estimated number next month.

REPORT 8: Produce one cumulative aggregate report that relates to the maintenance information.



SQL | All Rows Fetched: 25 in 0.022 seconds

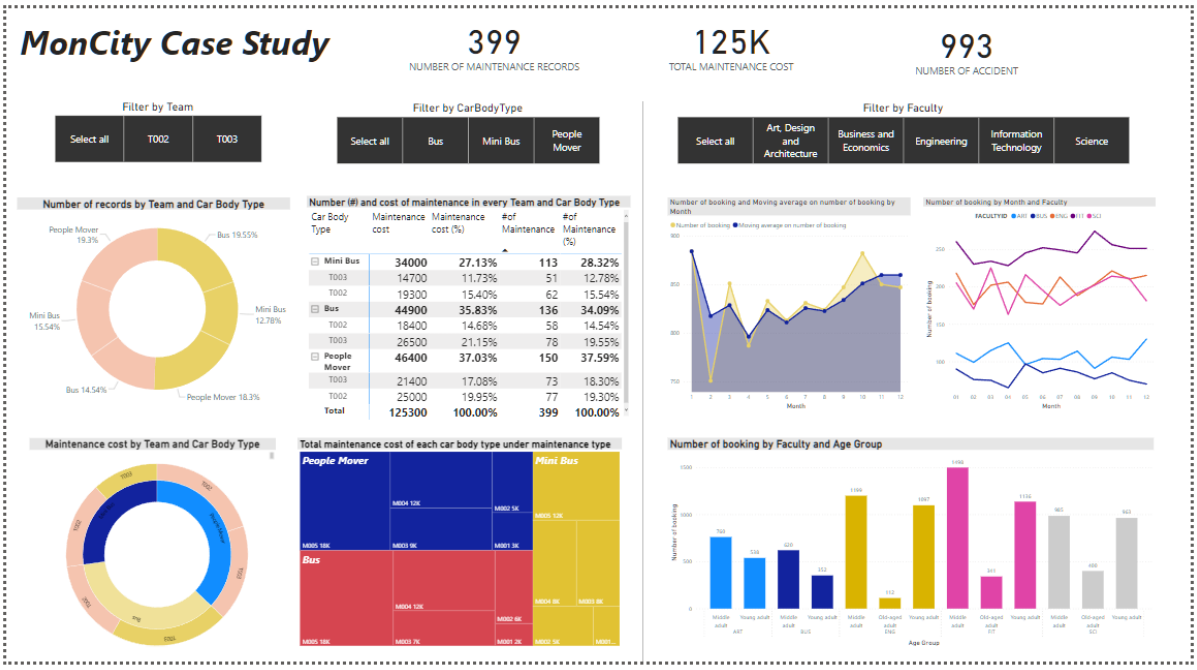
	MAINTENANCETYPE	TEAMID	TOTAL_NUM_OF_MAIN_RECORD	TOTAL_MAIN_COST	CUM_TOTAL_MAIN_COST
1	M001	T001	45	4,500	4,500
2	M001	T002	31	3,100	7,600
3	M001	T003	38	3,800	11,400
4	M001	T004	47	4,700	16,100
5	M001	T005	43	4,300	20,400
6	M002	T001	46	9,200	29,600
7	M002	T002	38	7,600	37,200
8	M002	T003	39	7,800	45,000
9	M002	T004	29	5,800	50,800

```
-- Display the cumulative total cost used based on maintenance type,
-- and another cumulative total used for each team.
SELECT maintenancetype, teamid,
SUM(num_of_main_record) AS total_num_of_main_record,
to_char(SUM(main_cost), '9,999,999,999') AS total_main_cost,
TO_CHAR (SUM(SUM(main_cost)) OVER
(ORDER BY maintenancetype, teamid
ROWS UNBOUNDED PRECEDING), '9,999,999,999') AS cum_total_main_cost
FROM maintenancefact_v1
GROUP BY maintenancetype, teamid;
```

This query is valuable because we can use cumulative total cost to estimate total expenditures from all team cumulatively. This is a evidence to convince the top management that the project is investable.

4. Business Intelligence (BI) Reports

The following is the dashboard built from report 2, report 4, report 5, report 7.

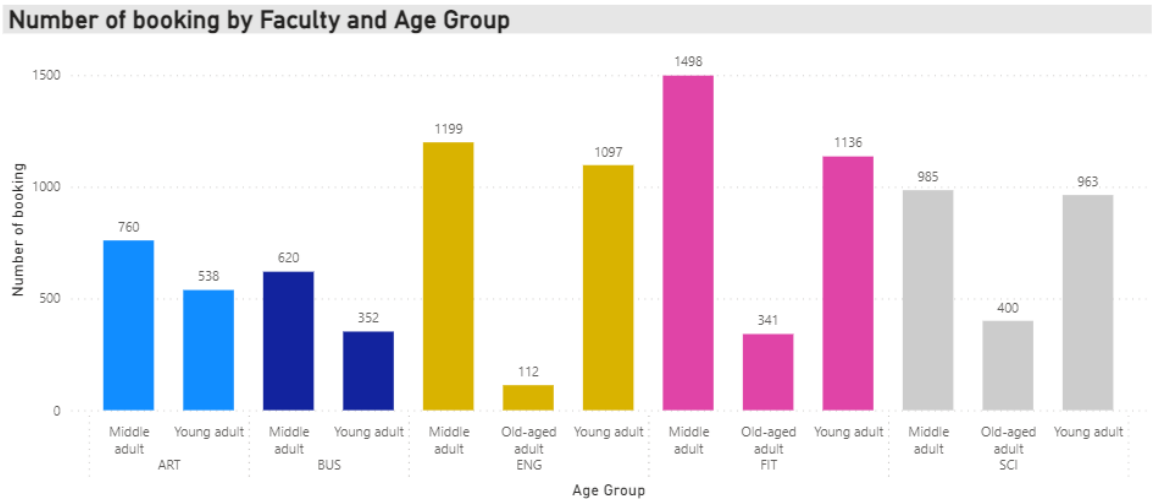


The dashboard is published here.

5. Final Recommendations/Suggestions

To support business intelligence needs, part of my jobs is to design data warehouse based on business objectives and develop reports with insight of customer behavior and help identify trends in business performance over time. As such, here makes four suggestion to our manager and other relevant stakeholders like Data / Business Analyst.

Suggestion 1: Marketing campaigns specifically for old-age adult group



As MonCity has an increasing trend on aging population and generally old-aged groups are in needs of a more convenient way of commute, we think that there is a demand for this group of people. This graph shows that the demographic of old-age adult group contributes the lowest number of booking across all faculty. To expand our customer base, we should target this clientele. Generally, offering discounts, having affiliated marketing, and email marketing campaign are the examples of marketing campaigns. This is to gain the awareness of our brand, and knowing that our service could help them Therefore, a part of the investment should go towards the marketing on old-aged group.

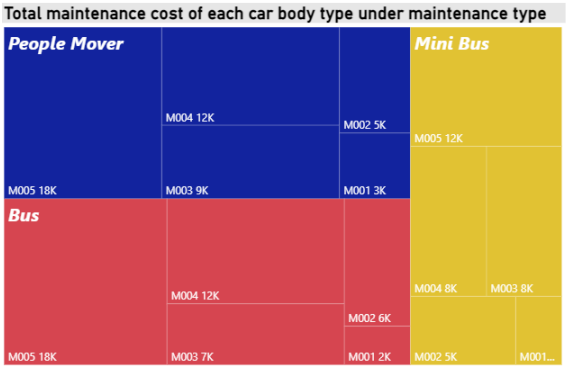
Suggestion 2: Buying more bus

Number (#) and cost of maintenance in every Team and Car Body Type

Car Body Type	Maintenance cost	Maintenance cost (%)	#of Maintenance	#of Maintenance (%)
Mini Bus	**34000**	**27.13%**	**113**	**28.32%**
T003	14700	11.73%	51	12.78%
T002	19300	15.40%	62	15.54%
Bus	**44900**	**35.83%**	**136**	**34.09%**
T002	18400	14.68%	58	14.54%
T003	26500	21.15%	78	19.55%
People Mover	**46400**	**37.03%**	**150**	**37.59%**
T003	21400	17.08%	73	18.30%
T002	25000	19.95%	77	19.30%
Total	**125300**	**100.00%**	**399**	**100.00%**

In expanding our business, we will need buy more cars, and this pivot table can give an insight to decide which should be bought. This figure shows that People Mover has higher maintenance cost but less seats (10), whereas Minibus has lower maintenance cost but more seats (20). From a pragmatic point of view, we want to buy cars that have less maintenance cost meanwhile containing more seats to take more passengers. Therefore, when deciding which types, we would not recommend buying more People mover because of high maintenance cost with the least number of seats. While the maintenance cost of bus is not as low as mini bus but but it can carry twice as much. We hence suggest that the further expenditure on buying cars should go towards buying bus.

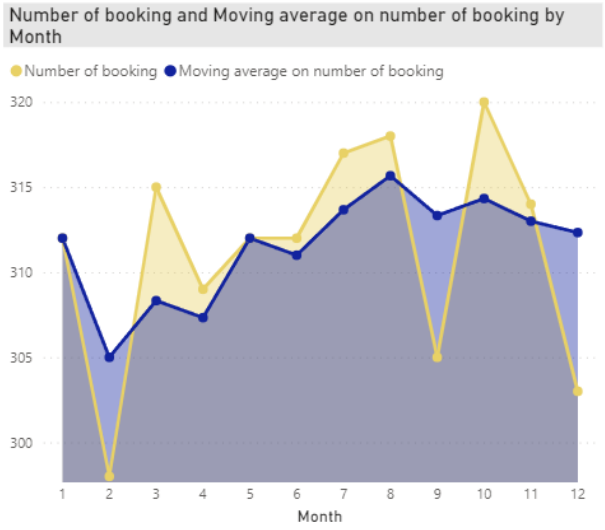
Suggestion 3: New KPI should be made in tracking the performance of maintenance cost

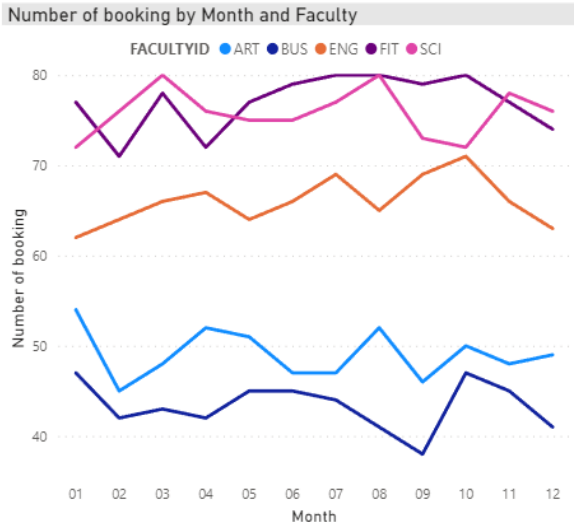


With cumulative maintenance cost being 125,000, we should worry about the cost of maintenance. To reach a breakeven, we suggest that Data Analyst and other top management should work together and come up with a new KPI in tracking the cost.

From the left chart, we found that M005 has contributed the most maintenance cost, therefore, such a KPI needs to take a greater account for M005, and find a way to minimise the type of maintenance.

Suggestion 4: Promotions on slack months or seasons





We suggest that there could be a promotion on a specific month or seasons. For example, February has the has lower number of booking maybe because that was the holiday. To boost the sales, we suggest the marketing team could investigate that further.

In conclusion, the above suggestions — including inflow and outflow of the investment — are based on the dashboard we made. For further analysis or a need to edit the dashboard, please feel free to contact our team.