FIT3003 Major Assignment 2022

Student Name: Amy Wang June Koh, Cherline Delfina Tandra
Student ID: 29796601, 31864767
Last Date Modified: 12th October 2022

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
29796601	KOH	AMY WANG JUNE
31864767	TANDRA	CHERLINE DELFINA

Unit name and code	FIT3003 BUSINESS INTELLIGENCE AND DATA WAREHOUSE	
Title of assignment	FIT3003 MAJOR ASSIGNMENT 2022	
Lecturer/tutor	DR. AGNES HARYANTO	
Tutorial day and time	TUESDAY 6PM – 8PM	**Campus: CLAYTON CAMPUS**
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: 12TH OCTOBER 2022		**Date submitted: 12TH OCTOBER 2022**

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 The material used can be from any source (staff, students or the internet, published and unpublished works).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.

* delete (iii) if not applicable

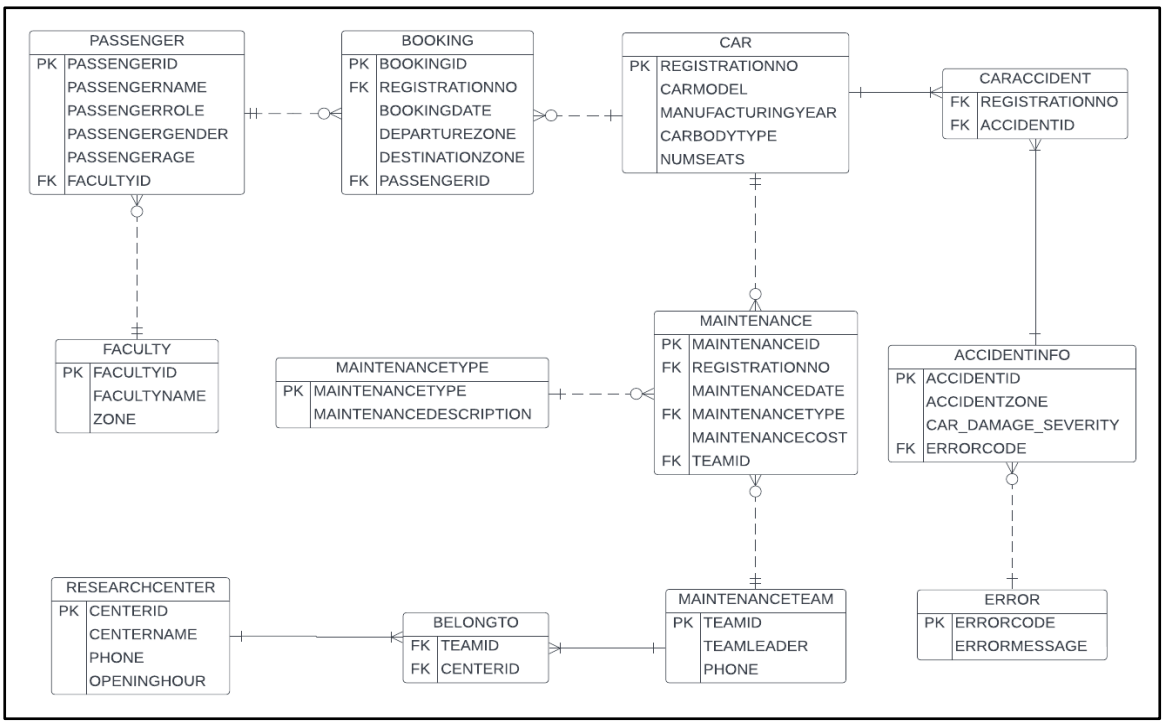
Signature ___amykoh_____ Date: 12th October 2022___ Signature _____cherlinetandra_____ Date:__12th October 2022___

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

Table of Contents

Task C.1

A. Entity Relationship Diagram (ERD)



Link for the clearer version of ERD is provided in the appendix.

B. Data Cleaning Process

1. Relationship Problem: Illegal FacultyId (‘Alienware’) in Passenger table not in Faculty Table.

```
-- CHECK
SELECT *
FROM MONCITY.PASSENGER
WHERE FACULTYID NOT IN (SELECT FACULTYID FROM   MONCITY.FACULTY);

-- FIX
DROP TABLE PASSENGER CASCADE CONSTRAINTS PURGE;
CREATE TABLE PASSENGER AS
 SELECT *
 FROM MONCITY.PASSENGER
 WHERE FACULTYID IN (SELECT FACULTYID FROM   MONCITY.FACULTY);
```

PASSENGERID	PASSENGERNAME	PASSENGERROLE	PASSENGERGENDER	PASSENGERAGE	FACULTYID
U163	Anabia Mccabe	Staff	Male	21	Alienware

FACULTYID
FIT
BUS
ENG
ART
SCI

PASSENGERID	PASSENGERNAME	PASSENGERROLE	PASSENGERGENDER	PASSENGERAGE	FACULTYID
U020	Hope Shepard	Student	Female	42	FIT
U028	Serenity Herring	Staff	Female	69	FIT
U036	Tess Sheppard	Staff	Female	65	FIT
U040	Xzavier Robles	Student	Female	35	FIT
U041	Roberto Oliver	Student	Female	32	FIT
U042	Alyssa Shepherd	Staff	Male	48	FIT

2. Incorrect Values: MaintenanceId M2000 have maintenance cost -200, which it cannot be less than 0.

```
-- CHECK
SELECT *
FROM MONCITY.MAINTENANCE
WHERE MAINTENANCECOST < 0;

-- FIX
DROP TABLE MAINTENANCE CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCE AS
 SELECT *
 FROM   MONCITY.MAINTENANCE
 WHERE MAINTENANCECOST > 0;
```

MAINTENANCEID	REGISTRATIONNO	MAINTENANCEDATE	MAINTENANCETYPE	MAINTENANCECOST	TEAMID
M2000	Car13	19-JUL-15	M002	-200	T004

MAINTENANCEID	REGISTRATIONNO	MAINTENANCEDATE	MAINTENANCETYPE	MAINTENANCECOST	TEAMID
M200	Car18	08-AUG-15	M002	200	T005
M201	Car14	21-JAN-18	M001	100	T003
M202	Car10	08-AUG-20	M004	400	T005
M203	Car01	31-OCT-21	M002	200	T001
M204	Car03	19-NOV-16	M001	100	T001

3. Duplicate Problem: 2 BookingId, T1218 is found in the booking table.

```
-- CHECK
SELECT BOOKINGID, COUNT(*) AS DUPLICATE_RECORDS_COUNT
FROM MONCITY.BOOKING
GROUP BY BOOKINGID
HAVING COUNT (*) > 1;

-- FIX
DROP TABLE BOOKING CASCADE CONSTRAINTS PURGE;
CREATE TABLE BOOKING AS
  SELECT DISTINCT *
  FROM MONCITY.BOOKING;
```

BOOKINGID	DUPLICATE_RECORDS_COUNT
T1218	2

BOOKINGID	REGISTRATIONNO	BOOKINGDATE	DEPARTUREZONE	DESTINATIONZONE	PASSENGERID
T1218	Car14	29-JUL-18	ZoneD	ZoneB	U059

4. Null Value Problem: Null primary key AccidentId is found in AccidentInfo table.

```
-- CHECK
SELECT *
FROM MONCITY.ACCIDENTINFO
WHERE ACCIDENTID IS NULL;

-- FIX
DROP TABLE ACCIDENTINFO CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENTINFO AS
 SELECT *
 FROM MONCITY.ACCIDENTINFO
 WHERE ACCIDENTID IS NOT NULL;
```

ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
(null)	ZoneC	Severe damage	Error005

ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
A632	ZoneA	No damage	Error002
A633	ZoneB	No damage	Error002
A634	ZoneB	Severe damage	Error001
A635	ZoneD	Severe damage	Error001
A636	ZoneC	Severe damage	Error005

5. Relationship Problem: Illegal Errorcode (‘ERROR010’) in AccidentInfo table is not found in Error table.

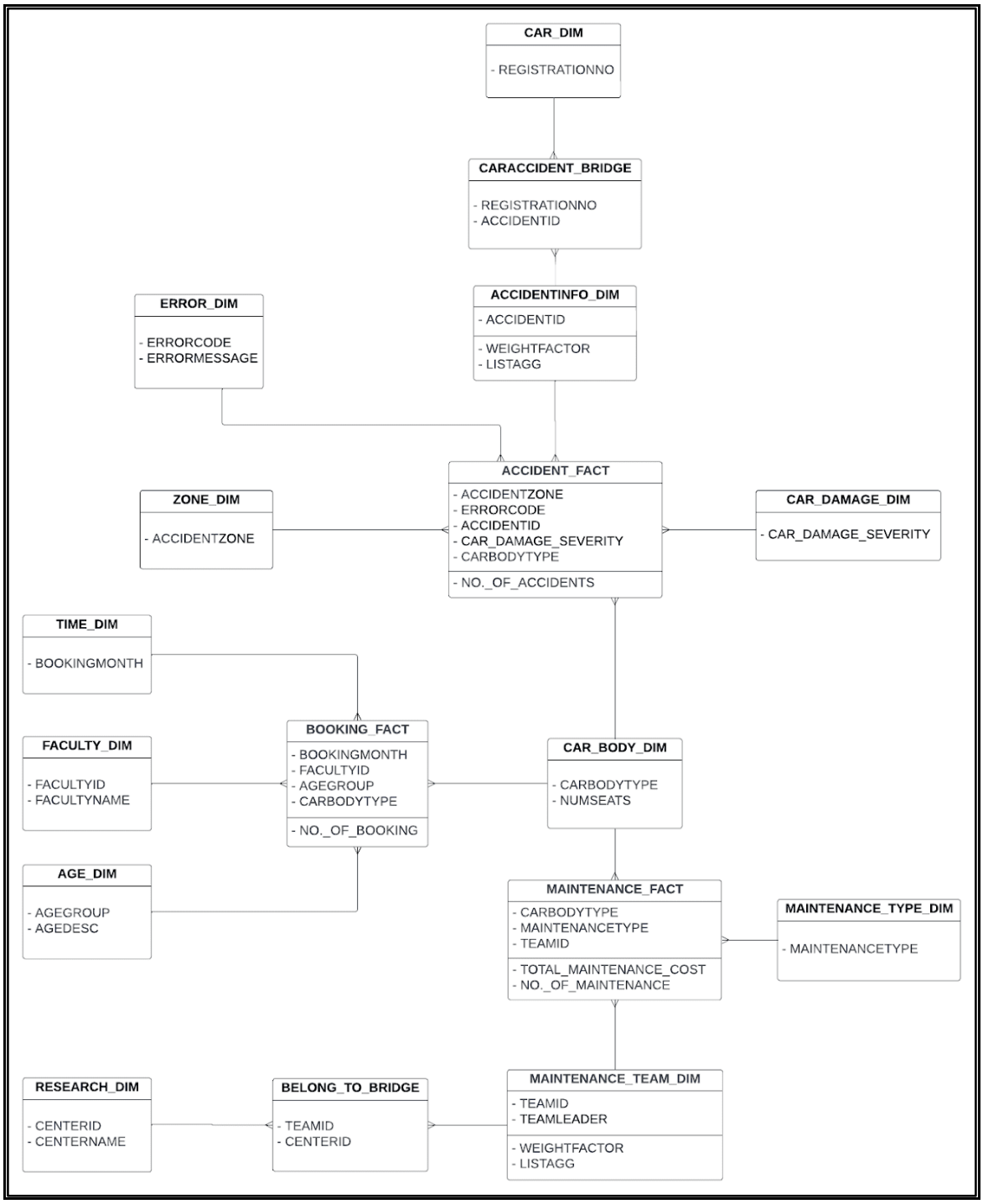
```
-- CHECK
SELECT *
FROM MONCITY.ACCIDENTINFO
WHERE ERRORCODE NOT IN (SELECT ERRORCODE
FROM MONCITY.ERROR);

-- FIX
DELETE FROM ACCIDENTINFO
WHERE ERRORCODE NOT IN (SELECT ERRORCODE
FROM MONCITY.ERROR);
```

ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
A2000	ZoneB	No damage	Error010

ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
A632	ZoneA	No damage	Error002
A633	ZoneB	No damage	Error002
A634	ZoneB	Severe damage	Error001
A635	ZoneD	Severe damage	Error001
A636	ZoneC	Severe damage	Error005

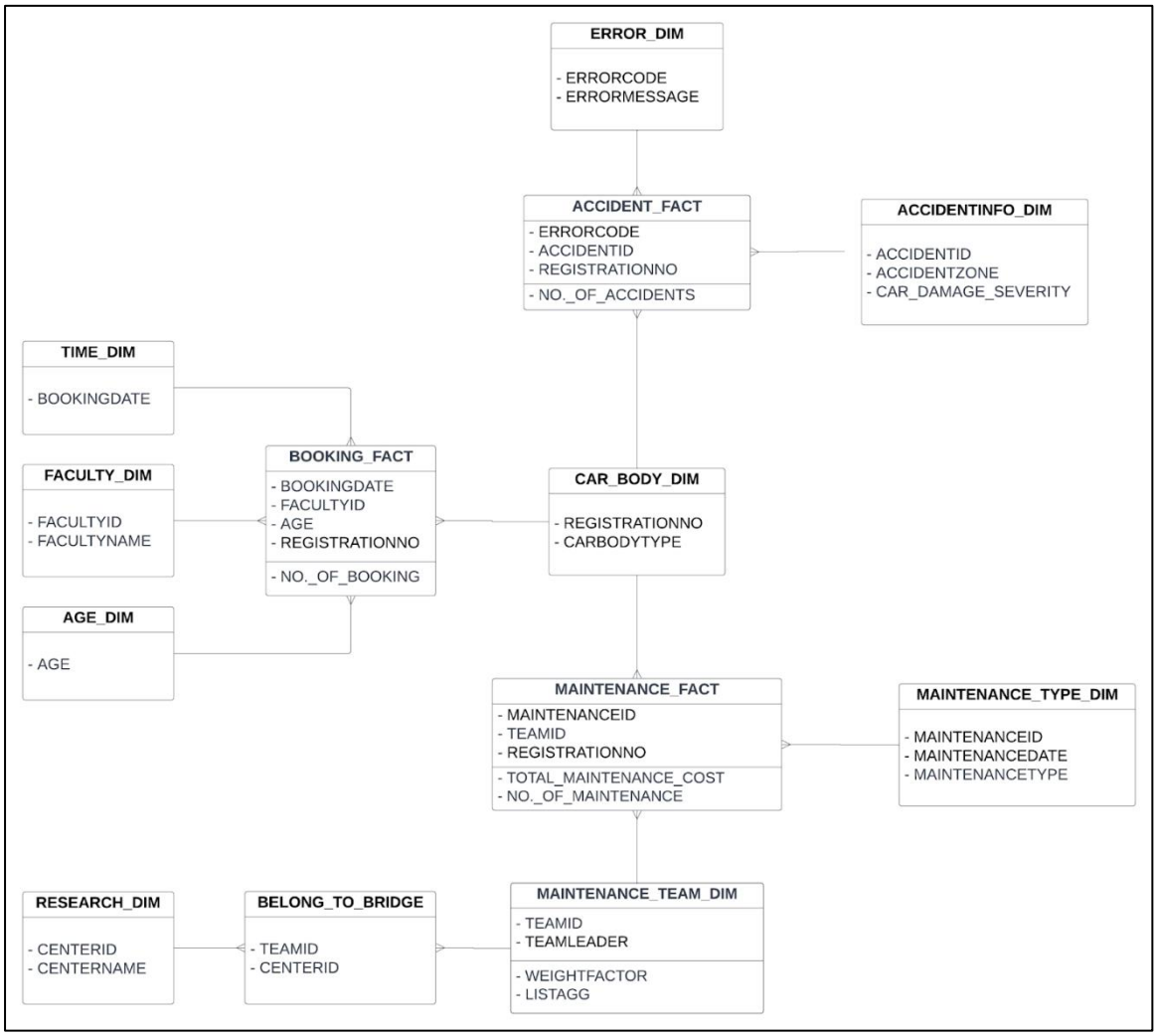
C. Star Schema

Version 1



Link for the clearer version of Star Schema V1 is provided in the appendix.

Version 2



Link for the clearer version of Star Schema V2 is provided in the appendix.

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

Answer: Our start schema does not use the determinant dimension because there's no need to separate a row attribute into columns in the report, which is what determinant dimension is used for.

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

Answer: Temporal dimension is used in the level 1 aggregation to turn AGE into AGEGROUP. Type 0 SCD is used because it only records the initial passenger's age when the data warehousing is built.

E. Difference between Star Schema Version 1 and Version 2.

Answer: The difference between star schema version 1 and version 2 is that version 1 has a lower granularity of the fact measure than star schema version 2. As you can see the time and age dimension table had been grouped together instead of the original time and age values taken from the original table. Booking month is used in the time dimension table and age group has been used in the age dimension table. So, this version of the star schema will have fewer detailed data as the data had been aggregated or grouped together.

Moreover, the star schema version 2, obviously, will have a higher granularity compared to the star schema version 1. Star schema version 2 consists of the most detailed data. As you can see on the star schema version 2 above, there is no aggregation in the fact tables. It is almost identical to the entity relationship diagram we had done earlier on. The date and age dimension table are not grouped together. In other words, we replaced the existing date and age dimension with a higher granularity dimension. Thus, the value in the fact table will be broken down into more detailed data because the fact measure has a lower detail dimension. Furthermore, we also added more data into the maintenance dimension which are the maintenance id, maintenance date, and maintenance type. Accident Id also has been added to the accident info dimension, so that the accident fact table will have a more detailed value.

Task C.2

A. SQL Star Schema Version 1

```
-- CAR_DIM1
DROP TABLE CAR_DIM1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE CAR_DIM1 AS SELECT DISTINCT REGISTRATIONNO
FROM MONCITY.CAR;

-- CARACCIDENT_BRIDGE
DROP TABLE CARACCIDENT_BRIDGE CASCADE CONSTRAINTS PURGE;
CREATE TABLE CARACCIDENT_BRIDGE AS SELECT DISTINCT REGISTRATIONNO,
ACCIDENTID
FROM MONCITY.CARACCIDENT;

-- ACCIDENTINFO_DIM1
DROP TABLE ACCIDENTINFO_DIM1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENTINFO_DIM1 AS SELECT DISTINCT A.ACCIDENTID,
1.0/COUNT(*) AS WEIGHTFACTOR,
LISTAGG (CA.REGISTRATIONNO, '_') WITHIN GROUP
(ORDER BY CA.REGISTRATIONNO) AS REGISTRATIONGROUPLIST
FROM ACCIDENTINFO A, MONCITY.CARACCIDENT CA
WHERE A.ACCIDENTID = CA.ACCIDENTID
GROUP BY A.ACCIDENTID;

-- ERROR_DIM1
DROP TABLE ERROR_DIM1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ERROR_DIM1 AS
SELECT ERRORCODE, ERRORMESSAGE FROM MONCITY.ERROR;

-- ACCIDENT_FACT1
DROP TABLE ACCIDENT_FACT1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENT_FACT1
```

```
AS SELECT DISTINCT A.ACCIDENTID, A.ACCIDENTZONE, E.ERRORCODE,
C.CARBODYTYPE, A.CAR_DAMAGE_SEVERITY, COUNT(*) AS NO_OF_ACCIDENTS

FROM MONCITY.CAR C, ACCIDENTINFO A,  MONCITY.ERROR E,
MONCITY.CARACCIDENT CA

WHERE A.ACCIDENTID = CA.ACCIDENTID

AND c.REGISTRATIONNO = CA.REGISTRATIONNO

AND E.ERRORCODE = A.ERRORCODE

GROUP BY A.ACCIDENTZONE, A.ACCIDENTID, E.ERRORCODE,
A.CAR_DAMAGE_SEVERITY, C.CARBODYTYPE

ORDER BY A.ACCIDENTID;


-- ZONE_DIM1

DROP TABLE ZONE_DIM1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE ZONE_DIM1 AS SELECT DISTINCT ACCIDENTZONE

FROM ACCIDENTINFO;


-- CAR_DAMAGE_DIM1

DROP TABLE CAR_DAMAGE_DIM1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE CAR_DAMAGE_DIM1 AS SELECT DISTINCT CAR_DAMAGE_SEVERITY

FROM ACCIDENTINFO;


-- CAR_BODY_DIM1

DROP TABLE CAR_BODY_DIM1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE CAR_BODY_DIM1 AS SELECT DISTINCT CARBODYTYPE, NUMSEATS

FROM MONCITY.CAR;


-- TEMPFACT

DROP TABLE TEMPFACT1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE TEMPFACT1

AS SELECT TO_CHAR(B.BOOKINGDATE, 'Month') AS BOOKINGMONTH, F.FACULTYID,
P.PASSENGERAGE AS AGE, C.REGISTRATIONNO, C.CARBODYTYPE

FROM PASSENGER P, BOOKING B,  MONCITY.FACULTY F, MONCITY.CAR C

WHERE F.FACULTYID = P.FACULTYID

AND P.PASSENGERID = B.PASSENGERID
```

```
AND C.REGISTRATIONNO = B.REGISTRATIONNO

GROUP BY B.BOOKINGDATE, F.FACULTYID, P.PASSENGERAGE, C.CARBODYTYPE,
C.REGISTRATIONNO;


-- UPDATE TEMPFACT TO TURN AGE INTO AGE GROUP

ALTER TABLE TEMPFACT1

ADD (AGEGROUP VARCHAR2(30));


UPDATE TEMPFACT1

SET AGEGROUP = 'GROUP 1'

WHERE AGE >=18 AND AGE <=35;


UPDATE TEMPFACT1

SET AGEGROUP = 'GROUP 2'

WHERE AGE >=36 AND AGE <=59;


UPDATE TEMPFACT1

SET AGEGROUP = 'GROUP 3'

WHERE AGE >=60;


-- BOOKING_FACT1

DROP TABLE BOOKING_FACT1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE BOOKING_FACT1

AS SELECT BOOKINGMONTH, FACULTYID, AGEGROUP, CARBODYTYPE, COUNT(*) as
NO_OF_BOOKING

FROM TEMPFACT1

GROUP BY BOOKINGMONTH, FACULTYID, AGEGROUP, CARBODYTYPE;


-- TIME_DIM1

DROP TABLE TIME_DIM1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE TIME_DIM1 AS SELECT DISTINCT TO_CHAR(BOOKINGDATE, 'Month')
AS BOOKINGMONTH

FROM BOOKING;
```

```
-- FACULTY_DIM1
DROP TABLE FACULTY_DIM1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE FACULTY_DIM1 AS
SELECT DISTINCT FACULTYID, FACULTYNAME FROM MONCITY.FACULTY;

-- AGE_DIM1
DROP TABLE AGE_DIM1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE AGE_DIM1
(AGEGROUP VARCHAR2(30) NOT NULL,
 AGEDESC VARCHAR2(30),
 PRIMARY KEY(AGEGROUP));

INSERT INTO AGE_DIM1 VALUES ('18-35 YEARS OLD', 'YOUNG ADULTS');
INSERT INTO AGE_DIM1 VALUES ('36-59 YEARS OLD', 'MIDDLE-AGED ADULTS');
INSERT INTO AGE_DIM1 VALUES ('OVER 60 YEARS OLD', 'OLD-AGED ADULTS');

-- MAINTENANCE_FACT1
DROP TABLE MAINTENANCE_FACT1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCE_FACT1
AS SELECT c.CARBODYTYPE, mt.TEAMID, m.MAINTENANCETYPE, count(*) as
NO_OF_MAINTENANCE, sum(m.MAINTENANCECOST) AS TOTAL_MAINTENANCE_COST
FROM MONCITY.CAR c, MONCITY.MAINTENANCETEAM mt, MAINTENANCE m
WHERE c.REGISTRATIONNO = m.REGISTRATIONNO
AND m.TEAMID = mt.TEAMID
GROUP BY c.CARBODYTYPE, mt.TEAMID, m.MAINTENANCETYPE;

-- MAINTENANCE_TYPE_DIM1
DROP TABLE MAINTENANCE_TYPE_DIM1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCE_TYPE_DIM1 AS SELECT DISTINCT
MAINTENANCETYPE
FROM MAINTENANCE;
```

```
-- MAINTENANCE_TEAM_DIM1

DROP TABLE MAINTENANCE_TEAM_DIM1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE MAINTENANCE_TEAM_DIM1 AS SELECT DISTINCT M.TEAMID,
M.TEAMLEADER,

1.0/COUNT(*) AS WEIGHTFACTOR,

LISTAGG (B.CENTERID, '_') WITHIN GROUP

(ORDER BY B.CENTERID) AS CENTERGROUPLIST

FROM MONCITY.MAINTENANCETEAM M, MONCITY.BELONGTO B

WHERE M.TEAMID = B.TEAMID

GROUP BY M.TEAMID, M.TEAMLEADER;


-- BELONG_TO_BRIDGE

DROP TABLE BELONG_TO_BRIDGE CASCADE CONSTRAINTS PURGE;

CREATE TABLE BELONG_TO_BRIDGE AS

SELECT DISTINCT * FROM MONCITY.BELONGTO;


-- RESEARCH_DIM1

DROP TABLE RESEARCH_DIM1 CASCADE CONSTRAINTS PURGE;

CREATE TABLE RESEARCH_DIM1 AS

SELECT DISTINCT CENTERID, CENTERNAME FROM MONCITY.RESEARCHCENTER;


COMMIT;
```

B. SQL Star Schema Version 2

```
-- TIME_DIM2

DROP TABLE TIME_DIM2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE TIME_DIM2 AS SELECT DISTINCT BOOKINGDATE

FROM BOOKING;


-- AGE_DIM2

DROP TABLE AGE_DIM2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE AGE_DIM2 AS SELECT DISTINCT PASSENGERAGE AS AGE

FROM PASSENGER;


--ACCIDENTINFO_DIM2

DROP TABLE ACCIDENTINFO_DIM2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE ACCIDENTINFO_DIM2 AS SELECT DISTINCT A.ACCIDENTID,
A.ACCIDENTZONE, A.CAR_DAMAGE_SEVERITY

FROM ACCIDENTINFO A, MONCITY.CARACCIDENT CA

WHERE A.ACCIDENTID = CA.ACCIDENTID

GROUP BY A.ACCIDENTID, A.ACCIDENTZONE, A.CAR_DAMAGE_SEVERITY;


-- MAINTENANCE_TYPE_DIM2

DROP TABLE MAINTENANCE_TYPE_DIM2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE MAINTENANCE_TYPE_DIM2 AS SELECT DISTINCT MAINTENANCEID,
MAINTENANCEDATE, MAINTENANCETYPE

FROM MAINTENANCE;


-- ACCIDENT_FACT2

DROP TABLE ACCIDENT_FACT2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE ACCIDENT_FACT2

AS SELECT A.ACCIDENTID, E.ERRORCODE, CA.REGISTRATIONNO, COUNT(*) AS
NO_OF_ACCIDENTS

FROM ACCIDENTINFO A, MONCITY.ERROR E, MONCITY.CARACCIDENT CA

WHERE E.ERRORCODE = A.ERRORCODE
```

```
AND A.ACCIDENTID = CA.ACCIDENTID

GROUP BY A.ACCIDENTID, E.ERRORCODE, CA.REGISTRATIONNO;


-- CAR_BODY_DIM2

DROP TABLE CAR_BODY_DIM2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE CAR_BODY_DIM2 AS SELECT DISTINCT REGISTRATIONNO,
CARBODYTYPE, NUMSEATS

FROM MONCITY.CAR;


-- BOOKING_FACT2

DROP TABLE BOOKING_FACT2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE BOOKING_FACT2

AS SELECT B.BOOKINGDATE, F.FACULTYID, P.PASSENGERAGE AS AGE,
C.REGISTRATIONNO, COUNT(*) AS NO_OF_BOOKING

FROM PASSENGER P, BOOKING B,  MONCITY.FACULTY F, MONCITY.CAR C

WHERE F.FACULTYID = P.FACULTYID

AND P.PASSENGERID = B.PASSENGERID

AND C.REGISTRATIONNO = B.REGISTRATIONNO

GROUP BY B.BOOKINGDATE, F.FACULTYID, P.PASSENGERAGE, C.REGISTRATIONNO;


-- MAINTENANCE_FACT2

DROP TABLE MAINTENANCE_FACT2 CASCADE CONSTRAINTS PURGE;

CREATE TABLE MAINTENANCE_FACT2

AS SELECT C.REGISTRATIONNO, MT.TEAMID, M.MAINTENANCEID, COUNT(*) AS
NO_OF_MAINTENANCE, SUM(M.MAINTENANCECOST) AS
TOTAL_MAINTENANCE_COST

FROM MONCITY.CAR C, MONCITY.MAINTENANCETEAM MT, MAINTENANCE M

WHERE C.REGISTRATIONNO = M.REGISTRATIONNO

AND M.TEAMID = MT.TEAMID

GROUP BY C.REGISTRATIONNO, MT.TEAMID, M.MAINTENANCEID;


COMMIT;
```

Task C.3

Report 1

```
SELECT FACULTYID AS "FacultyID", BOOKINGMONTH AS "Month",
SUM(NO_OF_BOOKING) AS "Total bookings",

SUM(SUM(NO_OF_BOOKING)) OVER (ORDER BY FACULTYID,
TO_DATE(BOOKINGMONTH, 'MONTH')) AS "Cumulative number of booking records"

FROM BOOKING_FACT1

WHERE FACULTYID LIKE 'FIT'

GROUP BY FACULTYID, BOOKINGMONTH

ORDER BY TO_DATE(BOOKINGMONTH, 'MONTH');
```

FacultyID	Month	Total bookings	Cumulative number of booking records
FIT	January	260	260
FIT	February	230	490
FIT	March	234	724
FIT	April	228	952
FIT	May	245	1197
FIT	June	252	1449
FIT	July	249	1698
FIT	August	245	1943
FIT	September	274	2217
FIT	October	256	2473
FIT	November	251	2724
FIT	December	251	2975

Report 2

```
SELECT DECODE(GROUPING(M.TEAMID),1,'All Teams', M.TEAMID) AS "Team ID",
DECODE(GROUPING(C.CARBODYTYPE),1,'All Car Body Types', C.CARBODYTYPE) AS "Car body type",
SUM(M.NO_OF_MAINTENANCE) AS "Total number of maintenance",
TO_CHAR (SUM(M.TOTAL_MAINTENANCE_COST),'9,999,999,999') AS "Total maintenance cost"
FROM MAINTENANCE_FACT1 M, CAR_BODY_DIM1 C
WHERE M.CARBODYTYPE = C.CARBODYTYPE AND
M.TEAMID IN ('T002','T003')
GROUP BY CUBE(M.TEAMID, C.CARBODYTYPE)
ORDER BY "Team ID", "Car body type";
```

Team ID	Car body type	Total number of maintenance	Total maintenance cost
All Teams	All Car Body Types	399	125,300
All Teams	Bus	136	44,900
All Teams	Mini Bus	113	34,000
All Teams	People Mover	150	46,400
T002	All Car Body Types	197	62,700
T002	Bus	58	18,400
T002	Mini Bus	62	19,300
T002	People Mover	77	25,000
T003	All Car Body Types	202	62,600
T003	Bus	78	26,500

Report 3

```
SELECT DISTINCT A.ERRORCODE AS "Error Code", REGISTRATIONNO AS "Registration No.",

A.CARBODYTYPE AS "Car body type", A.TOTAL_NUMBER_OF_ACCIDENTS AS "Total number of accidents", A."Rank" FROM (

SELECT A.ERRORCODE, CAR.REGISTRATIONNO, C.CARBODYTYPE, SUM(NO_OF_ACCIDENTS) AS TOTAL_NUMBER_OF_ACCIDENTS,

DENSE_RANK() OVER(PARTITION BY A.ERRORCODE ORDER BY SUM(NO_OF_ACCIDENTS)DESC) AS "Rank"

FROM ACCIDENT_FACT1 A, CAR_BODY_DIM1 C, ACCIDENTINFO_DIM1 AI, CARACCIDENT_BRIDGE CA, CAR_DIM CAR

WHERE A.CARBODYTYPE = C.CARBODYTYPE

AND AI.ACCIDENTID = A.ACCIDENTID

AND CA.ACCIDENTID = AI.ACCIDENTID

AND CA.REGISTRATIONNO = CAR.REGISTRATIONNO

GROUP BY A.ERRORCODE, CAR.REGISTRATIONNO, C.CARBODYTYPE) A, ACCIDENTINFO AI

WHERE A.ERRORCODE = AI.ERRORCODE

AND (UPPER(AI.CAR_DAMAGE_SEVERITY) = 'VERY MINOR DAMAGE' OR UPPER(A.ERRORCODE) = 'ERROR002')

AND A."Rank" <=3

ORDER BY A.ERRORCODE, A."Rank";
```

Error Code	Registration No.	Car body type	Total number of accidents	Rank
Error001	Car01	Bus	13	1
Error001	Car04	Bus	12	2
Error001	Car12	Mini Bus	12	2
Error001	Car19	Mini Bus	12	2
Error001	Car08	Bus	11	3
Error001	Car20	Mini Bus	11	3
Error002	Car22	People Mover	45	1
Error002	Car27	People Mover	42	2
Error002	Car23	People Mover	39	3
Error002	Car30	People Mover	39	3
Error003	Car06	Bus	12	1

Report 4

```
SELECT C.CARBODYTYPE AS "Car body type",
DECODE(GROUPING(B.AGEGROUP),1,'All Age Groups', B.AGEGROUP) AS "Age group",
DECODE(GROUPING(B.FACULTYID),1,'All Faculties', INITCAP(B.FACULTYID)) AS "Faculty ID",
TO_CHAR(SUM(NO_OF_BOOKING),'9,999,999,999') AS "Total number of bookings"
FROM BOOKING_FACT1 B, CAR_BODY_DIM1 C
WHERE B.CARBODYTYPE = C.CARBODYTYPE AND
UPPER(C.CARBODYTYPE) LIKE 'PEOPLE MOVER'
GROUP BY C.CARBODYTYPE, ROLLUP(B.AGEGROUP), ROLLUP(B.FACULTYID)
ORDER BY "Age group", "Faculty ID", SUM(NO_OF_BOOKING) DESC;
```

Car body type	Age group	Faculty ID	Total number of bookings
People Mover	All Age Groups	All Faculties	3,396
People Mover	All Age Groups	Art	453
People Mover	All Age Groups	Bus	314
People Mover	All Age Groups	Eng	841
People Mover	All Age Groups	Fit	1,009
People Mover	All Age Groups	Sci	779
People Mover	GROUP 1	All Faculties	1,380
People Mover	GROUP 1	Art	169
People Mover	GROUP 1	Bus	121
People Mover	GROUP 1	Eng	382

Report 5

```
SELECT DECODE(GROUPING(BOOKINGMONTH),1,'All Months', BOOKINGMONTH) AS "Months",
DECODE(GROUPING(FACULTYID),1,'All Faculties', FACULTYID) AS "Faculties",
TO_CHAR(SUM(NO_OF_BOOKING),'9,999,999,999') AS "Total Bookings"
FROM BOOKING_FACT1
GROUP BY ROLLUP(BOOKINGMONTH, FACULTYID)
ORDER BY TO_DATE(BOOKINGMONTH, 'MONTH'), "Total Bookings" DESC, "Faculties";
```

Months	Faculties	Total Bookings
JANUARY	All Faculties	884
JANUARY	FIT	260
JANUARY	ENG	218
JANUARY	SCI	205
JANUARY	ART	111
JANUARY	BUS	90
FEBRUARY	All Faculties	751
FEBRUARY	FIT	230
FEBRUARY	ENG	176
FEBRUARY	SCI	170

Report 6

```
SELECT DECODE(GROUPING(AGEGROUP),1,'ALL AGE', AGEGROUP) AS "Age Group",
FACULTYID AS "Faculties",
TO_CHAR(SUM(NO_OF_BOOKING),'9,999,999,999') AS "Total Bookings"
FROM BOOKING_FACT1
GROUP BY ROLLUP(AGEGROUP), FACULTYID
ORDER BY AGEGROUP, "Total Bookings" DESC, "Faculties";
```

Age Group	Faculties	Total Bookings
GROUP 1	FIT	1,136
GROUP 1	ENG	1,097
GROUP 1	SCI	963
GROUP 1	ART	537
GROUP 1	BUS	352
GROUP 2	FIT	1,498
GROUP 2	ENG	1,199
GROUP 2	SCI	985
GROUP 2	ART	760
GROUP 2	BUS	620

An explanation of the differences between rollup and partial rollup

Answer: The difference between rollup and partial rollup is that rollup gets a set of attribute names that need to be grouped together in order to produce the subtotals of rolling-up aggregate combinations of the specified attributes given and the grand total of them. Whereas partial rollup only takes into account the specified attributes given, and produces the total of that given attribute together with another group by attributes.

Report 7

```
SELECT BOOKINGMONTH AS "Months", C.CARBODYTYPE AS "Car body type",
ROUND(SUM(NO_OF_BOOKING)) AS  "Total bookings",

ROUND(AVG(SUM(NO_OF_BOOKING)) OVER (ORDER BY C.CARBODYTYPE,
BOOKINGMONTH ROWS 2 PRECEDING)) AS "Moving 3 Months Average"

FROM BOOKING_FACT1 B, CAR_BODY_DIM1 C

WHERE B.CARBODYTYPE = C.CARBODYTYPE

GROUP BY BOOKINGMONTH, C.CARBODYTYPE

ORDER BY TO_DATE(BOOKINGMONTH, 'MONTH');
```

Months	Car body type	Total bookings	Moving 3 Months Average
January	Mini Bus	293	260
January	Bus	291	283
January	People Mover	300	284
February	Mini Bus	211	255
February	People Mover	261	280
February	Bus	279	272
March	Bus	260	262
March	People Mover	302	296
March	Mini Bus	289	274
April	Mini Bus	264	275
April	Bus	250	250
April	People Mover	273	294

Report 8

```
SELECT C.CARBODYTYPE AS "Car body type", M.MAINTENANCETYPE AS "Maintenance type",
SUM(TOTAL_MAINTENANCE_COST) AS "Total maintenance cost",
SUM(SUM(TOTAL_MAINTENANCE_COST)) OVER (ORDER BY C.CARBODYTYPE, M.MAINTENANCETYPE) AS "Cumulative number of maintenance cost"
FROM MAINTENANCE_FACT1 M, CAR_BODY_DIM1 C
WHERE M.CARBODYTYPE = C.CARBODYTYPE
GROUP BY C.CARBODYTYPE, M.MAINTENANCETYPE
ORDER BY C.CARBODYTYPE, M.MAINTENANCETYPE;
```

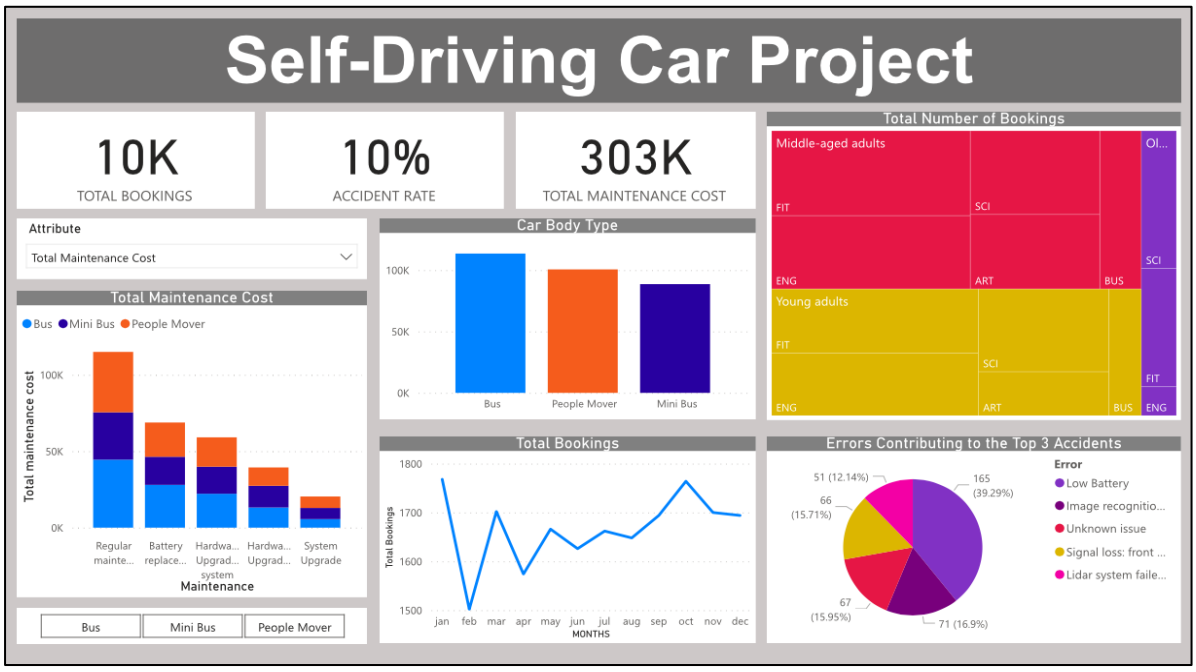
Car body type	Maintenance type	Total maintenance cost	Cumulative number of maintenance cost
Bus	M001	5600	5600
Bus	M002	13200	18800
Bus	M003	22200	41000
Bus	M004	28000	69000
Bus	M005	44500	113500
Mini Bus	M001	7300	120800
Mini Bus	M002	14200	135000
Mini Bus	M003	17700	152700
Mini Bus	M004	18400	171100
Mini Bus	M005	31000	202100
People Mover	M001	7500	209600
People Mover	M002	12000	221600
People Mover	M003	19200	240800
People Mover	M004	22400	263200
People Mover	M005	39500	302700

An explanation on why such a query is necessary or valuable for management

Answer: Query allows us to see trends and summary, it allows the management team to make rational decisions based on it. For instance, by accumulating the total cost of the maintenance, we can get the overall total maintenance cost for all the car body type and maintenance type. Thus, the management team will know how much the company earned without the need of calculating them one by one. By accumulating the total bookings per month, it allows the management team to see the trends on which month has higher / lower demand and manage inventory to keep up with demand. By accumulating the total accidents and error code, it allows the management team to see which error caused the most accidents, which they can then report to the company to fix the error.

Task C. 4

Power BI Report



Task C.5

The business intelligence dashboard was built to help the management decide on self-driving car projects. It has been clearly stated that the accident rate for transportation is about 10 over 100 percent, which means that about 10 out of 100 cards will get into an accident. Thus, we can say that the self-driving car project is not worth further investment because it is dangerous and unprofitable. Secondly, based on the dashboard, the total maintenance fees for 10k of transport will be around 303k. This means they will need to charge 30k per booking just to break even with the maintenance fees, excluding the production cost, marketing cost, etc. Therefore, we can say that it is way high over the budget, cost-ineffective and probably the company will have to invest a lot more money into this project before even publishing them to the people.

However, if the management of Moncity is adamant about pursuing this self-driving project, they may need to make several changes to lower the business cost. Firstly, as you can see on the dashboard that mentioned errors contributing to the top 3 accidents, it had been clearly stated almost 39% of accidents are caused by low battery. Therefore, Moncity should have increased the transport's battery capacity. The image recognition system should also be improved as it will cause severe incidents if it is not resolved. Moncity should have hired some experts in the Artificial Intelligence field so that they can solve this issue immediately.

Furthermore, the target audience for Moncity should be focused on middle-aged adults whose age is between 39 and 59 years old. It is because they contributed the most bookings based on the database. In addition, we know that January and October have the highest booking out of all the other months while February has the lowest booking. Therefore, Moncity should organise and manage their inventory well to make sure they have enough stock in January and October to keep up with demand.

For the transportation sites, it is not ideal for Moncity to invest in the people mover and bus as they have the highest number of accidents contributed even though they have the highest number of bookings. On the other hand, the minibus can be considered by the Moncity as it has the cheapest maintenance cost and second highest on the bookings. However, it is dangerous too because it ranks second in the number of accidents.

In conclusion, if Moncity does decide to pursue a self-driving project, they should invest more in minibuses instead of buses and people movers. Huge changes should be made by Moncity to ensure the safety of the people and the costs of the business. Nevertheless, if Moncity decides not to make any changes to the project, the self-driving car is not safe enough to be put out to the community, and the project will not be profitable which may cause the company to go down the pan.

Appendix

ERD Link

https://lucid.app/lucidchart/9e20943a-e10f-4f98-8edd-f2dc54e33841/edit?viewport_loc=-10%2C-10%2C1707%2C768%2C0_0&invitationId=inv_7a45d4f0-eb19-4117-8ce7-d410da114c3d#

Star Schema Link

https://lucid.app/lucidchart/15ed71a6-9994-408a-9942-4e5a02609320/edit?viewport_loc=-3030%2C-1285%2C7240%2C3442%2C0_0&invitationId=inv_a231b583-bb5a-4016-b4a5-8e3adab831e8#

Contribution Declaration

Student ID	**Student Name**	**Contribution Percentage (%)**
29796601	Amy Wang June Koh	50%
31864767	Cherline Delfina Tandra	50%

TASK DONE BY INDIVIDUAL	
Student ID: 29796601	Student ID: 31864767
1. Task C.1 a. 80% ERD Diagram b. 70% Star Schema Version 1 Diagram c. 20% Star Schema Version 2 Diagram d. 70% SQL Star Schema Version 1 e. 40% SQL Star Schema Version 2 2. Task C.2 a. 70% SQL Star Schema Version 1 and Version 2 3. 90% of Task C.3 4. 50% Task C.5	1. Task C.1 a. 20% ERD Diagram b. Data cleaning c. 30% Star Schema Version 1 Diagram d. 80% Star Schema Version 2 Diagram e. 30% SQL Star Schema Version 1 f. 60% SQL Star Schema Version 2 2. Task C.2 a. 30% SQL Star Schema Version 1 and Version 2 3. 10% of Task C.3 4. Task C.4 5. 50% Task C.5