FIT3003 Business Intelligence and Data Warehousing

Major Assignment: BI & DW

Lai Qui Juin - 32638809
Lee Bao Qi - 31855237

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
31855237	LEE	BAO QI
32638809	LAI	QUI JUIN

* Please include the names of all other group members.

Unit name and code	FIT3003: Business Intelligence and Data Warehousing	
Title of assignment	Major Assignment: BI & DW	
Lecturer/tutor	Dr. Soon Lay Ki	
Tutorial day and time	Tuesday 8am	**Campus** Malaysia
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 12/10/2022		**Date submitted** 12/10/2022

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 The material used can be from any source (staff, students or the internet, published and unpublished works).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

Signature ..... *Date*.....

* delete (iii) if not applicable

Signature Lai Qui Juin Date: 10/10/2022

Signature Lee Bao Qi Date: 10/10/2022

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

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
31855237	Lee Bao Qi	50%
32638809	Lai Qui Juin	50%

Student ID: 31855237	Student ID: 32638809

- Task C.1
  - Preparation stage (outputs a, b)
  - data cleaning (Strategy 3 and 4)
  - ER Diagram
  - Star/snowflake schema v1 and v2 (output c)
  - Explanation and differences (output d and e)
- Task C.2
  - SQL for Star/snowflake schema (output b)
- Task C.3 OLAP Queries and Report with moving and cumulative aggregate
  - Reports 3 and 4 (outputs a and b)
  - Reports 7 and 8 (outputs a, b, c, d)
- Task C.4 Business Intelligence (BI) Reports
  - Reports 3 and 4
  - Design BI Report
- Task C.5 Final Recommendation / Suggestions
	- Task C.1
	  - Preparation stage (outputs a, b)
	  - data cleaning (Strategy 1 and 2)
	  - ER Diagram
	  - Star/snowflake schema v1 and v2 (output c)
	  - Explanation and differences (output d and e)
	- Task C.2
	  - SQL for Star/snowflake schema (output a)
	- Task C.3 OLAP Queries and Reports will rollup and partial rollup
	  - Reports 1 and 2 (outputs a and b)
	  - Reports 5 and 6 (outputs a, b, c and d)
	- Task C.4 Business Intelligence (BI) Reports
	  - Reports 2 and 6
	  - Design BI Report
	- Task C.5 Final Recommendation / Suggestions

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	Lai Qui Juin	Lee Bao Qi
Date	10 / 10 / 2022 (Day Month Year)	

Table of Contents

C. Tasks

1. Data Warehouse Preparation and Design

1.1. Preparation Stage

1.1.1. E/R Diagram

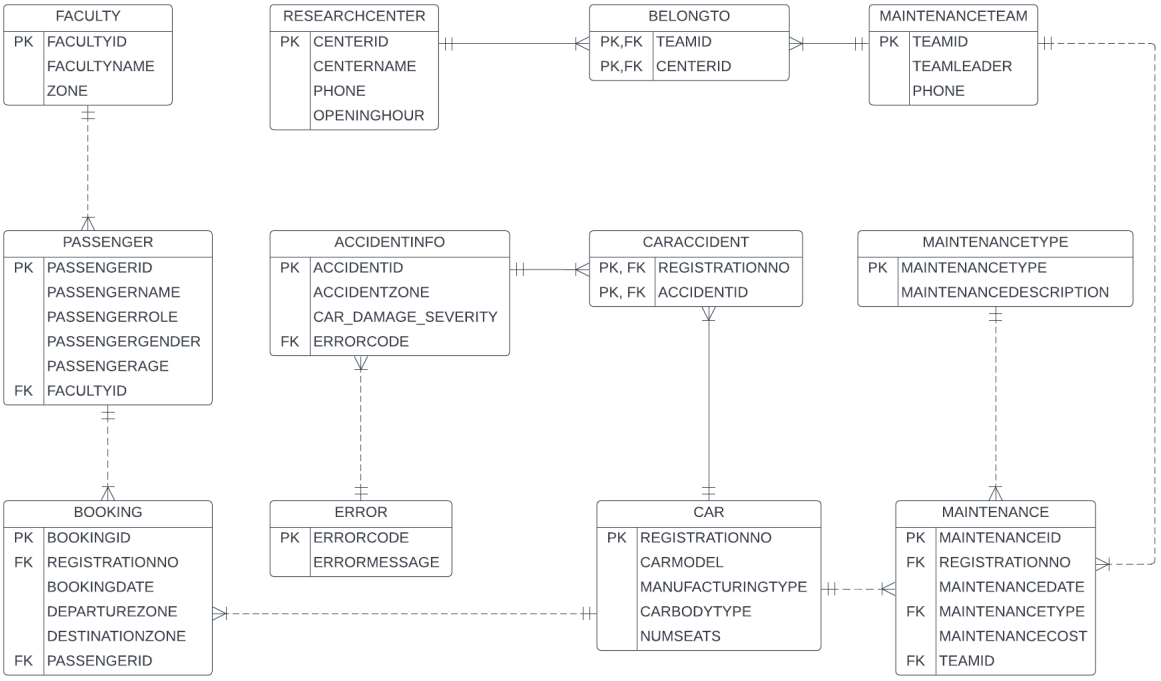

Figure 1. E/R Diagram of the Operational Database of MonCity

1.1.2. Tables Creation

```
-- creating tables by copying the tables from operational database
DROP TABLE FACULTY CASCADE CONSTRAINTS PURGE;
CREATE TABLE FACULTY AS
SELECT * FROM MONCITY.FACULTY;

DROP TABLE RESEARCHCENTER CASCADE CONSTRAINTS PURGE;
CREATE TABLE RESEARCHCENTER AS
SELECT * FROM MONCITY.RESEARCHCENTER;

DROP TABLE PASSENGER CASCADE CONSTRAINTS PURGE;
CREATE TABLE PASSENGER AS
SELECT * FROM MONCITY.PASSENGER;

DROP TABLE ERROR CASCADE CONSTRAINTS PURGE;
```

```
CREATE TABLE ERROR AS
SELECT * FROM MONCITY.ERROR;

DROP TABLE MAINTENANCETYPE CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCETYPE AS
SELECT * FROM MONCITY.MAINTENANCETYPE;

DROP TABLE BOOKING CASCADE CONSTRAINTS PURGE;
CREATE TABLE BOOKING AS
SELECT * FROM MONCITY.BOOKING;

DROP TABLE CAR CASCADE CONSTRAINTS PURGE;
CREATE TABLE CAR AS
SELECT * FROM MONCITY.CAR;

DROP TABLE MAINTENANCE CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCE AS
SELECT * FROM MONCITY.MAINTENANCE;

DROP TABLE ACCIDENTINFO CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENTINFO AS
SELECT * FROM MONCITY.ACCIDENTINFO;

DROP TABLE CARACCIDENT CASCADE CONSTRAINTS PURGE;
CREATE TABLE CARACCIDENT AS
SELECT * FROM MONCITY.CARACCIDENT;

DROP TABLE MAINTENANCETEAM CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCETEAM AS
SELECT * FROM MONCITY.MAINTENANCETEAM;

DROP TABLE BELONGTO CASCADE CONSTRAINTS PURGE;
CREATE TABLE BELONGTO AS
SELECT * FROM MONCITY.BELONGTO;
```

1.1.3. Data Exploration and Cleaning

```
-- data exploration
SELECT * FROM FACULTY;
SELECT * FROM RESEARCHCENTER;
SELECT * FROM PASSENGER;
SELECT * FROM ERROR;
SELECT * FROM MAINTENANCETYPE;
SELECT * FROM BOOKING;
SELECT * FROM CAR;
SELECT * FROM MAINTENANCE;
SELECT * FROM ACCIDENTINFO;
```

```
SELECT * FROM CARACCIDENT;
SELECT * FROM MAINTENANCETEAM;
SELECT * FROM BELONGTO;
```

-- Strategies for data exploration and cleaning

To explore the operational database and perform data cleaning, we used 4 strategies:

1. Check if there are any invalid records that exist in one table but not in another
2. Check if there are duplicate records
3. Check if there are negative records (cost, age, numseats)
4. Check if there are null values in non-nullable columns (such as PK, FK)

-- 1. Check if there are any invalid records that exist in one table but not in another

```
--check if there are invalid records in FACULTY table
SELECT * FROM FACULTY;

-- check if there are invalid records in RESEARCHCENTER table;
SELECT * FROM RESEARCHCENTER;

-- check if there are invalid records in PASSENGER table;
SELECT * FROM PASSENGER;
SELECT * FROM PASSENGER ORDER BY PASSENGERID;
SELECT PASSENGERROLE, COUNT(*) FROM PASSENGER GROUP BY
PASSENGERROLE;
SELECT PASSENGERGENDER, COUNT(*) FROM PASSENGER GROUP BY
PASSENGERGENDER;
SELECT * FROM PASSENGER WHERE FACULTYID NOT IN (SELECT FACULTYID FROM
FACULTY); --error found!
```

-- ERROR 1: Faculty id for passenger U163 does not exist in the faculty table

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

```
-- To fix this, delete this record from the PASSENGER table since it does not affect other
tables
SELECT * FROM BOOKING WHERE PASSENGERID = 'U163';
DELETE FROM PASSENGER WHERE PASSENGERID = 'U163';
-- check the table again
SELECT * FROM PASSENGER WHERE FACULTYID NOT IN (SELECT FACULTYID FROM
FACULTY);
```

PASSENGERID	PASSENGERNAME	PASSENGERROLE	PASSENGERGENDER	PASSENGERAGE	FACULTYID

```
-- check if there are invalid records in ERROR table;
SELECT * FROM ERROR;

-- check if there are invalid records in MAINTENANCETYPE table;
```

```
SELECT * FROM MAINTENANCETYPE;

-- check if there are invalid records in BOOKING table;
SELECT * FROM BOOKING;
SELECT * FROM BOOKING WHERE REGISTRATIONNO NOT IN (SELECT
REGISTRATIONNO FROM CAR);
SELECT * FROM BOOKING WHERE DEPARTUREZONE NOT IN (SELECT ZONE FROM
FACULTY);
SELECT * FROM BOOKING WHERE DESTINATIONZONE NOT IN (SELECT ZONE FROM
FACULTY);
SELECT * FROM BOOKING WHERE PASSENGERID NOT IN (SELECT PASSENGERID
FROM PASSENGER);

-- check if there are invalid records in CAR table;
SELECT * FROM CAR;
SELECT CARBODYTYPE, COUNT(*) FROM CAR GROUP BY CARBODYTYPE;

-- check if there are invalid records in MAINTENANCE table;
SELECT * FROM MAINTENANCE ORDER BY MAINTENANCEID;
SELECT * FROM MAINTENANCE WHERE REGISTRATIONNO NOT IN (SELECT
REGISTRATIONNO FROM CAR);
SELECT * FROM MAINTENANCE WHERE MAINTENANCETYPE NOT IN (SELECT
MAINTENANCETYPE FROM MAINTENANCETYPE);
SELECT * FROM MAINTENANCE WHERE TEAMID NOT IN (SELECT TEAMID FROM
MAINTENANCETEAM);

-- check if there are invalid records in ACCIDENTINFO table;
SELECT * FROM ACCIDENTINFO ORDER BY ACCIDENTID;
SELECT * FROM ACCIDENTINFO WHERE ACCIDENTZONE NOT IN (SELECT ZONE
FROM FACULTY);
SELECT CAR_DAMAGE_SEVERITY, COUNT(*) FROM ACCIDENTINFO GROUP BY
CAR_DAMAGE_SEVERITY;
SELECT * FROM ACCIDENTINFO WHERE ERRORCODE NOT IN (SELECT
ERRORCODE FROM ERROR); -- error found!
```

-- ERROR 2: Invalid accident id 'A2000' found in ACCIDENTINFO table

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
1	A2000	ZoneB	No damage	Error010

```
-- To fix this, delete this record from the ACCIDENTINFO table since it does not affect other
tables
SELECT * FROM CARACCIDENT WHERE ACCIDENTID = 'A2000';
DELETE FROM ACCIDENTINFO WHERE ACCIDENTID = 'A2000';
-- check the table again
SELECT * FROM ACCIDENTINFO WHERE ERRORCODE NOT IN (SELECT
ERRORCODE FROM ERROR);
```

ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE

```
-- check if there are invalid records in CARACCIDENT table;
SELECT * FROM CARACCIDENT;
SELECT * FROM CARACCIDENT WHERE REGISTRATIONNO NOT IN (SELECT
REGISTRATIONNO FROM CAR);
SELECT * FROM CARACCIDENT WHERE ACCIDENTID NOT IN (SELECT ACCIDENTID
FROM ACCIDENTINFO);

-- check if there are invalid records in MAINTENANCETEAM table;
SELECT * FROM MAINTENANCETEAM;

-- check if there are invalid records in BELONGTO table;
SELECT * FROM BELONGTO;
SELECT * FROM BELONGTO WHERE TEAMID NOT IN (SELECT TEAMID FROM
MAINTENANCETEAM);
SELECT * FROM BELONGTO WHERE CENTERID NOT IN (SELECT CENTERID FROM
RESEARCHCENTER);
```

-- 2. Check if there are duplicate records

```
--check if there are duplicate records in FACULTY table;
SELECT FACULTYID, COUNT(*) FROM FACULTY GROUP BY FACULTYID HAVING
COUNT(*)>1;

-- check if there are duplicate records in RESEARCHCENTER table;
SELECT CENTERID, COUNT(*) FROM RESEARCHCENTER GROUP BY CENTERID
HAVING COUNT(*)>1;

-- check if there are duplicate records in PASSENGER table;
SELECT PASSENGERID, COUNT(*) FROM PASSENGER GROUP BY PASSENGERID
HAVING COUNT(*)>1;

-- check if there are duplicate records in ERROR table;
SELECT ERRORCODE, COUNT(*) FROM ERROR GROUP BY ERRORCODE HAVING
COUNT(*)>1;

-- check if there are duplicate records in MAINTENANCETYPE table;
SELECT MAINTENANCETYPE, COUNT(*) FROM MAINTENANCETYPE GROUP BY
MAINTENANCETYPE HAVING COUNT(*)>1;

-- check if there are duplicate records in BOOKING table;
SELECT BOOKINGID, COUNT(*) FROM BOOKING GROUP BY BOOKINGID HAVING
COUNT(*)>1; --error found!
```

-- ERROR 3: Two duplicated booking ids ‘T1218’ in BOOKING table

	BOOKINGID	COUNT(*)
1	T1218	2

```
-- To fix this, delete this record from the BOOKING table since it does not affect other tables
```

	ROWID
1	AABrR6AAHAABwwoACe
2	AABrR6AAHAABwzZAAE

	MIN(ROWID)
1	AABrR6AAHAABwwoACe

```
DELETE FROM BOOKING
WHERE BOOKINGID='T1218' AND
   ROWID NOT IN (
      SELECT MIN(ROWID)
      FROM BOOKING
      WHERE BOOKINGID='T1218');
-- check the table again
SELECT BOOKINGID, COUNT(*) FROM BOOKING GROUP BY BOOKINGID HAVING
COUNT(*)>1;
```

	BOOKINGID	COUNT(*)

```
-- check if there are duplicate records in CAR table;
SELECT REGISTRATIONNO, COUNT(*) FROM CAR GROUP BY REGISTRATIONNO
HAVING COUNT(*)>1;

-- check if there are duplicate records in MAINTENANCE table;
SELECT MAINTENANCEID, COUNT(*) FROM MAINTENANCE GROUP BY
MAINTENANCEID HAVING COUNT(*)>1;

-- check if there are duplicate records in ACCIDENTINFO table;
SELECT ACCIDENTID, COUNT(*) FROM ACCIDENTINFO GROUP BY ACCIDENTID
HAVING COUNT(*)>1;

-- check if there are duplicate records in MAINTENANCETEAM table;
SELECT TEAMID, COUNT(*) FROM MAINTENANCETEAM GROUP BY TEAMID HAVING
COUNT(*)>1;
```

-- 3. Check if there are negative records (cost, age, numseats)

```
-- check if there are any invalid passenger age
SELECT * FROM PASSENGER WHERE PASSENGERAGE < 0;

-- check if there are any invalid manufacturing year
SELECT * FROM CAR WHERE MANUFACTURINGYEAR < 0;

-- check if there are any invalid number of seats
SELECT * FROM CAR WHERE NUMSEATS < 0;

-- check if there are any invalid maintenance cost
SELECT * FROM MAINTENANCE WHERE MAINTENANCECOST < 0; -- error found!
```

	MAINTENANCEID	REGISTRATIONNO	MAINTENANCEDATE	MAINTENANCETYPE	MAINTENANCECOST	TEAMID
1	M2000	Car13	19/07/2015	M002	-200	T004

-- ERROR 4: Maintenance cost for Maintenance ID ‘M2000’ is a negative value

-- To fix this, delete this record from the MAINTENANCE table since it does not affect other tables
DELETE FROM MAINTENANCE WHERE MAINTENANCECOST < 0;
-- Check the table again
SELECT * FROM MAINTENANCE WHERE MAINTENANCECOST < 0;

-- 4. Check if there are null values in non-nullable columns (such as PK, FK)

-- check if there are any null faculty ID in FACULTY table
SELECT * FROM FACULTY WHERE FACULTYID IS NULL;

-- check if there are any null center ID in RESEARCHCENTER table
SELECT * FROM RESEARCHCENTER WHERE CENTERID IS NULL;

-- check if there are any null Passenger ID in PASSENGER table
SELECT * FROM PASSENGER WHERE PASSENGERID IS NULL;

-- check if there are any null Faculty ID in PASSENGER table
SELECT * FROM PASSENGER WHERE FACULTYID IS NULL;

-- check if there are any null Error code in ERROR table
SELECT * FROM ERROR WHERE ERRORCODE IS NULL;

-- check if there are any null Maintenance Type in MAINTENANCETYPE table
SELECT * FROM MAINTENANCETYPE WHERE MAINTENANCETYPE IS NULL;

-- check if there are any null Booking ID in BOOKING table
SELECT * FROM BOOKING WHERE BOOKINGID IS NULL;

-- check if there are any null Registration number in BOOKING table
SELECT * FROM BOOKING WHERE REGISTRATIONNO IS NULL;

-- check if there are any null Passenger ID in BOOKING table
SELECT * FROM BOOKING WHERE PASSENGERID IS NULL;

-- check if there are any null Registration number in CAR table
SELECT * FROM CAR WHERE REGISTRATIONNO IS NULL;

-- check if there are any null Maintenance ID in MAINTENANCE table
SELECT * FROM MAINTENANCE WHERE MAINTENANCEID IS NULL;

-- check if there are any null Registration number in MAINTENANCE table
SELECT * FROM MAINTENANCE WHERE REGISTRATIONNO IS NULL;

-- check if there are any null Maintenance Date in MAINTENANCE table

```
SELECT * FROM MAINTENANCE WHERE MAINTENANCEDATE IS NULL;

-- check if there are any null Maintenance Type in MAINTENANCE table
SELECT * FROM MAINTENANCE WHERE MAINTENANCETYPE IS NULL;

-- check if there are any null Team ID in MAINTENANCE table
SELECT * FROM MAINTENANCE WHERE TEAMID IS NULL;

-- check if there are any null Accident ID in ACCIDENTINFO table
SELECT * FROM ACCIDENTINFO WHERE ACCIDENTID IS NULL; -- Error found!
```

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
1	(null)	ZoneC	Severe damage	Error005

```
-- ERROR 5: Accident ID is a null value
-- To fix this, delete this record from the ACCIDENTINFO table since it does not appear in other tables that uses it such as CARACCIDENT
DELETE FROM ACCIDENTINFO WHERE ACCIDENTID IS NULL;
-- Check the table again
SELECT * FROM ACCIDENTINFO WHERE ACCIDENTID IS NULL;
```

ACCID...	ACCID...	CAR_D...	ERROR...

```
-- check if there are any null Error code in ACCIDENTINFO table
SELECT * FROM ACCIDENTINFO WHERE ERRORCODE IS NULL;

-- check if there are any null Registration number in CARACCIDENT table
SELECT * FROM CARACCIDENT WHERE REGISTRATIONNO IS NULL;

-- check if there are any null Accident ID in CARACCIDENT table
SELECT * FROM CARACCIDENT WHERE ACCIDENTID IS NULL;

-- check if there are any null Team ID in MAINTENANCETEAM table
SELECT * FROM MAINTENANCETEAM WHERE TEAMID IS NULL;

-- check if there are any null Team ID in BELONGTO table
SELECT * FROM BELONGTO WHERE TEAMID IS NULL;

-- check if there are any null Center ID in BELONGTO table
SELECT * FROM BELONGTO WHERE CENTERID IS NULL;
```

1.2. Star/Snowflake Schema Design

1.2.1. Star/Snowflake Schema Version-1

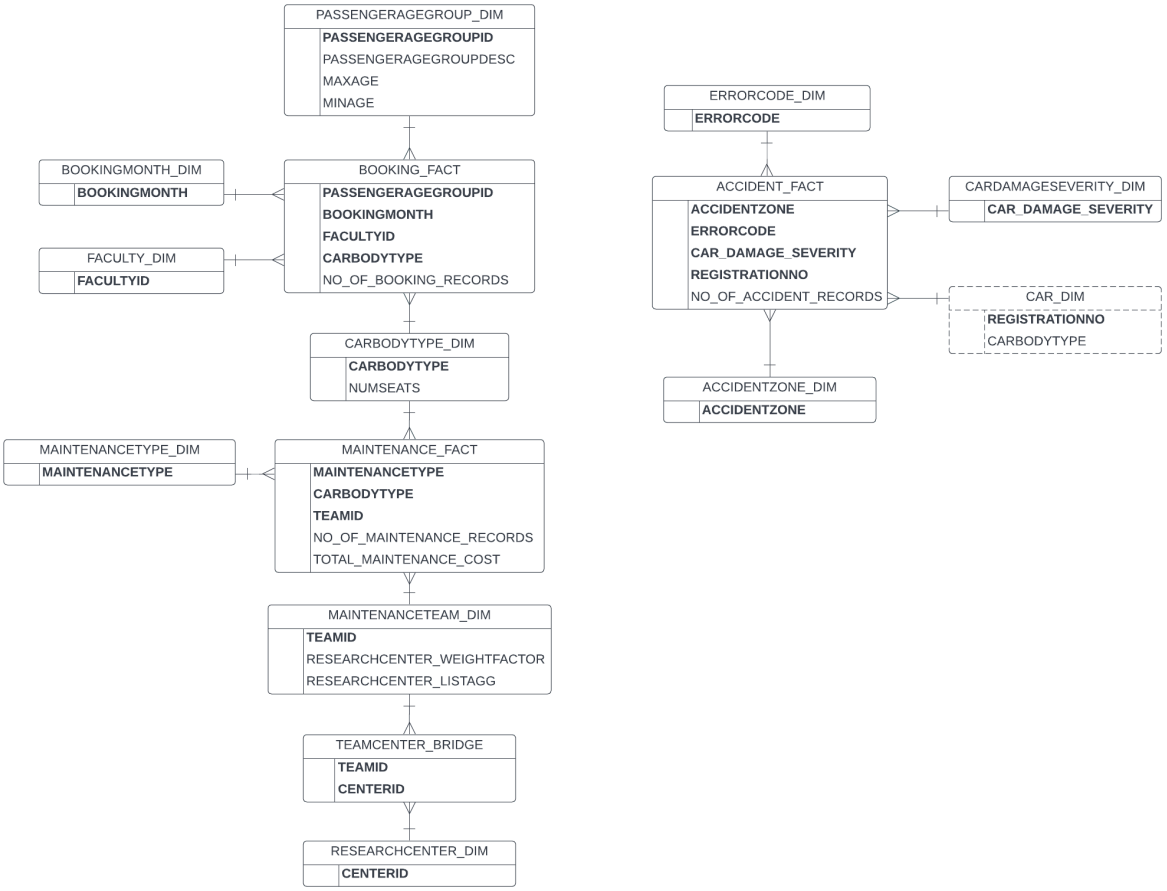

Figure 2. Star/Snowflake Schema Version-1 Diagram

1.2.2. Star/Snowflake Schema Version-2

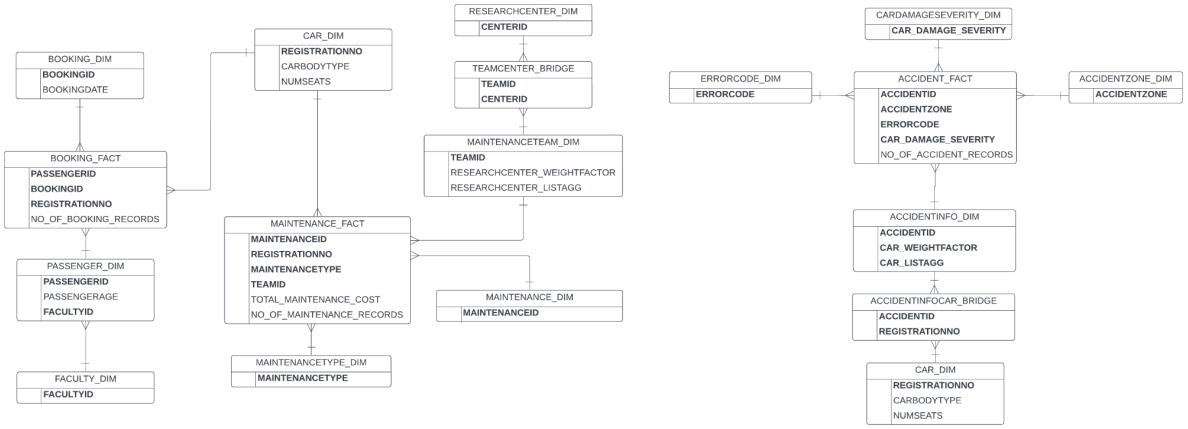

Figure 3. Star/Snowflake Schema Version-2 Diagram

1.2.3. Determinant Dimensions

For version-1, we have decided to include the Car dimension as a determinant dimension as the assignment specification stated that they are particularly interested in the number of accidents per self-driving car, so it is critical to use the car registration number in the queries in order to make the retrieved data meaningful.

For version-2, we have decided to not include determinant dimensions as fact measures are supposed to be non-aggregated.

1.2.4. Temporal Dimensions

The main reason for not implementing any temporal dimension was that there was no need for historical data to be tracked and there are no records within the dimensions that have a specific lifespan.

1.2.5. Differences between Star/Snowflake Schema Version-1 and Star/Snowflake Schema Version-2

In Level 0, booking_dim is created instead of bookingmonth_dim to store the actual booking date and its ID instead of the booking month to indicate the actual booking. passenger_dim is also introduced instead of passengeragegroup_dim to store the actual passenger age and their ID as well as faculty ID instead of the passenger's age group only. Passenger ID is used to help identify the passengers individually. The relationship between booking, passenger and faculty is represented as a hierarchy in order to follow the operational database which means the tables are normalized. There exists one-to-many relationships between faculty and passenger dimension, and between passenger dimension and booking fact table. maintenance_dim is added to prevent aggregation because one car can undergo more than one maintenance and thus have more than one maintenance ID to keep track of each individual maintenance record. car_dim is also created rather than using carbodytype_dim to identify individual cars. accidentinfo_dim is also included to get the accident ID to avoid aggregation by identifying each accident record.

For version-2, a bridge table between car_dim and accidentinfo_dim is also included because of the many-to-many relationship, as one car can have many accident records and one accident might involve more than one car. If the car dimension is linked directly to the accident fact table, it will cause aggregation in the number of accidents. There is also an absence of determinant dimension for cars in Level 0 as fact measures are supposed to be non-aggregated. For version-1, the car_dim is included as a determinant dimension as the assignment specification stated that they are particularly interested in the number of accidents per self-driving car, so it is critical to use the car registration number in the queries in order to make the retrieved data meaningful.

2. Star/Snowflake Schema Implementation

We decided to keep the two versions of star schema completely separated for simplicity and debugging purposes.

2.1. Star/Snowflake Schema Version-1

```
-- Creating Star Schema v1 (LEVEL 1)
-- Create car body type dimension
DROP TABLE CARBODYTYPE_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE CARBODYTYPE_DIM_V1 AS
   SELECT DISTINCT CARBODYTYPE, NUMSEATS FROM CAR;

SELECT * FROM CARBODYTYPE_DIM_V1;

-- Create faculty dimension
DROP TABLE FACULTY_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE FACULTY_DIM_V1 AS
   SELECT DISTINCT FACULTYID FROM PASSENGER;

SELECT * FROM FACULTY_DIM_V1;

-- Create passenger age group dimension
DROP TABLE PASSENGERAGEGROUP_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE PASSENGERAGEGROUP_DIM_V1 (
   PASSENGERAGEGROUPID   VARCHAR2(10),
   PASSENGERAGEGROUPDESC    VARCHAR2(50),
   MAXAGE    NUMBER,
   MINAGE      NUMBER
);

INSERT INTO PASSENGERAGEGROUP_DIM_V1 VALUES('Group1', 'Young adults', 35,
18);
INSERT INTO PASSENGERAGEGROUP_DIM_V1 VALUES('Group2', 'Middle-aged adults',
59, 36);
INSERT INTO PASSENGERAGEGROUP_DIM_V1 VALUES('Group3', 'Old-aged adults',
NULL, 60
);

SELECT * FROM PASSENGERAGEGROUP_DIM_V1;

-- Create booking month dimension
DROP TABLE BOOKINGMONTH_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE BOOKINGMONTH_DIM_V1 AS
   SELECT DISTINCT TO_CHAR(BOOKINGDATE, 'Month') AS BOOKINGMONTH
   FROM BOOKING
   ORDER BY TO_DATE(BOOKINGMONTH, 'Month');
```

```
SELECT * FROM BOOKINGMONTH_DIM_V1;

-- Create maintenance type dimension
DROP TABLE MAINTENANCETYPE_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCETYPE_DIM_V1 AS
   SELECT MAINTENANCETYPE FROM MAINTENANCETYPE;

SELECT * FROM MAINTENANCETYPE_DIM_V1;

-- Create research center dimension
DROP TABLE RESEARCHCENTER_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE RESEARCHCENTER_DIM_V1 AS
   SELECT CENTERID FROM RESEARCHCENTER;

SELECT * FROM RESEARCHCENTER_DIM_V1;

-- Create team center bridge table
DROP TABLE TEAMCENTER_BRIDGE_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE TEAMCENTER_BRIDGE_V1 AS
   SELECT * FROM BELONGTO;

SELECT * FROM TEAMCENTER_BRIDGE_V1;

-- Create maintenance team dimension
DROP TABLE MAINTENANCETEAM_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCETEAM_DIM_V1 AS
   SELECT T.TEAMID, ROUND(1.0/COUNT(B.CENTERID), 2) AS
RESEARCHCENTER_WEIGHTFACTOR,
         LISTAGG(B.CENTERID, ' ') WITHIN GROUP (ORDER BY B.CENTERID) AS
RESEARCHCENTER_LISTAGG
   FROM MAINTENANCETEAM T, BELONGTO B
   WHERE T.TEAMID=B.TEAMID
   GROUP BY T.TEAMID;

SELECT * FROM MAINTENANCETEAM_DIM_V1;

-- Create error code dimension
DROP TABLE ERRORCODE_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ERRORCODE_DIM_V1 AS
   SELECT ERRORCODE FROM ERROR;

SELECT * FROM ERRORCODE_DIM_V1;

-- Create accident zone dimension
DROP TABLE ACCIDENTZONE_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENTZONE_DIM_V1 AS
```

```
   SELECT DISTINCT ACCIDENTZONE FROM ACCIDENTINFO ORDER BY
ACCIDENTZONE;

SELECT * FROM ACCIDENTZONE_DIM_V1;

-- Create car damage severity dimension
DROP TABLE CARDAMAGESEVERITY_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE CARDAMAGESEVERITY_DIM_V1 AS
   SELECT DISTINCT CAR_DAMAGE_SEVERITY FROM ACCIDENTINFO;

SELECT * FROM CARDAMAGESEVERITY_DIM_V1;

-- Create car dimension
DROP TABLE CAR_DIM_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE CAR_DIM_V1 AS
   SELECT REGISTRATIONNO, CARBODYTYPE FROM CAR;

SELECT * FROM CAR_DIM_V1;

-- Create temporary Booking Fact table for star schema v-1 (level 1)
DROP TABLE TEMP_BOOKING_FACT_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE TEMP_BOOKING_FACT_V1 AS
   SELECT B.BOOKINGID, P.PASSENGERAGE, B.BOOKINGDATE, P.FACULTYID,
C.CARBODYTYPE
   FROM BOOKING B, CAR C, PASSENGER P
   WHERE B.PASSENGERID=P.PASSENGERID AND
       B.REGISTRATIONNO=C.REGISTRATIONNO;
SELECT * FROM TEMP_BOOKING_FACT_V1;

ALTER TABLE TEMP_BOOKING_FACT_V1
DROP COLUMN PASSENGERAGEGROUPID;

ALTER TABLE TEMP_BOOKING_FACT_V1
ADD (PASSENGERAGEGROUPID VARCHAR2(10));

UPDATE TEMP_BOOKING_FACT_V1
SET PASSENGERAGEGROUPID='Group1'
WHERE PASSENGERAGE>='18' AND
    PASSENGERAGE<='35';

UPDATE TEMP_BOOKING_FACT_V1
SET PASSENGERAGEGROUPID='Group2'
WHERE PASSENGERAGE>='36' AND
    PASSENGERAGE<='59';

UPDATE TEMP_BOOKING_FACT_V1
SET PASSENGERAGEGROUPID='Group3'
WHERE PASSENGERAGE>='60';
```

```
ALTER TABLE TEMP_BOOKING_FACT_V1
DROP COLUMN BOOKINGMONTH;

ALTER TABLE TEMP_BOOKING_FACT_V1
ADD (BOOKINGMONTH VARCHAR2(10));

UPDATE TEMP_BOOKING_FACT_V1
SET BOOKINGMONTH=TO_CHAR(BOOKINGDATE, 'Month');

SELECT * FROM TEMP_BOOKING_FACT_V1;

-- Create Booking Fact table for star schema v-1 (level 1)
DROP TABLE BOOKING_FACT_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE BOOKING_FACT_V1 AS
   SELECT TF.PASSENGERAGEGROUPID, P.PASSENGERAGEGROUPDESC,
TF.BOOKINGMONTH, TF.FACULTYID, TF.CARBODYTYPE, COUNT(TF.BOOKINGID) AS
NO_OF_BOOKING_RECORDS
   FROM TEMP_BOOKING_FACT_V1 TF, PASSENGERAGEGROUP_DIM_V1 P
   WHERE TF.PASSENGERAGEGROUPID=P.PASSENGERAGEGROUPID
   GROUP BY TF.PASSENGERAGEGROUPID, P.PASSENGERAGEGROUPDESC,
TF.BOOKINGMONTH, TF.FACULTYID, TF.CARBODYTYPE
   ORDER BY TF.CARBODYTYPE, TF.PASSENGERAGEGROUPID,
TO_DATE(TF.BOOKINGMONTH, 'MON');

SELECT * FROM BOOKING_FACT_V1;

-- Create Maintenance Fact table for star schema v-1 (level 1)
DROP TABLE MAINTENANCE_FACT_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCE_FACT_V1 AS
   SELECT M.TEAMID, C.CARBODYTYPE, M.MAINTENANCETYPE,
        COUNT(M.MAINTENANCEID) AS NO_OF_MAINTENANCE_RECORDS,
        SUM(M.MAINTENANCECOST) AS TOTAL_MAINTENANCE_COST
   FROM MAINTENANCE M, CAR C
   WHERE M.REGISTRATIONNO=C.REGISTRATIONNO
   GROUP BY M.TEAMID, C.CARBODYTYPE, M.MAINTENANCETYPE
   ORDER BY TEAMID;

SELECT * FROM MAINTENANCE_FACT_V1;

-- Create Accident Fact table for star schema v-1 (level 1)
DROP TABLE ACCIDENT_FACT_V1 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENT_FACT_V1 AS
   SELECT E.ERRORCODE, C.REGISTRATIONNO, C.CARBODYTYPE,
A.ACCIDENTZONE, A.CAR_DAMAGE_SEVERITY, COUNT(A.ACCIDENTID) AS
NO_OF_ACCIDENT_RECORDS
   FROM ACCIDENTINFO A, ERROR E, CAR C, CARACCIDENT CA
   WHERE A.ERRORCODE=E.ERRORCODE AND
```

```
        A.ACCIDENTID=CA.ACCIDENTID AND
        CA.REGISTRATIONNO=C.REGISTRATIONNO
   GROUP BY E.ERRORCODE, C.REGISTRATIONNO, C.CARBODYTYPE,
A.ACCIDENTZONE, A.CAR_DAMAGE_SEVERITY
   ORDER BY ERRORCODE, NO_OF_ACCIDENT_RECORDS DESC, ACCIDENTZONE,
CAR_DAMAGE_SEVERITY;

SELECT * FROM ACCIDENT_FACT_V1;
```

2.2. Star/Snowflake Schema Version-2

```
-- Creating Star Schema v2 (LEVEL 0)
-- Create booking dimension
DROP TABLE BOOKING_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE BOOKING_DIM_V2 AS
SELECT DISTINCT BOOKINGID, BOOKINGDATE
FROM BOOKING
ORDER BY BOOKINGID;

SELECT * FROM BOOKING_DIM_V2;

-- Create passenger dimension
DROP TABLE PASSENGER_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE PASSENGER_DIM_V2 AS
SELECT DISTINCT PASSENGERID, PASSENGERAGE, FACULTYID
FROM PASSENGER
ORDER BY PASSENGERID;

SELECT * FROM PASSENGER_DIM_V2;

-- Create faculty dimension
DROP TABLE FACULTY_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE FACULTY_DIM_V2 AS
SELECT DISTINCT FACULTYID
FROM FACULTY
ORDER BY FACULTYID;

SELECT * FROM FACULTY_DIM_V2;

-- Create car dimension
DROP TABLE CAR_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE CAR_DIM_V2 AS
SELECT DISTINCT REGISTRATIONNO, CARBODYTYPE, NUMSEATS
FROM CAR
ORDER BY REGISTRATIONNO;

SELECT * FROM CAR_DIM_V2;
```

```
-- Create booking fact for star schema v-2 (level 0)
DROP TABLE BOOKING_FACT_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE BOOKING_FACT_V2 AS
SELECT B.BOOKINGID, P.PASSENGERID, C.REGISTRATIONNO,
COUNT(B.BOOKINGID) AS NO_OF_BOOKING_RECORDS
FROM PASSENGER P, BOOKING B, CAR C
WHERE P.PASSENGERID = B.PASSENGERID
AND C.REGISTRATIONNO = B.REGISTRATIONNO
GROUP BY P.PASSENGERID, B.BOOKINGID, C.REGISTRATIONNO
ORDER BY B.BOOKINGID;

SELECT * FROM BOOKING_FACT_V2;

-- Create maintenance dimension
DROP TABLE MAINTENANCE_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCE_DIM_V2 AS
SELECT DISTINCT MAINTENANCEID
FROM MAINTENANCE
ORDER BY MAINTENANCEID;

SELECT * FROM MAINTENANCE_DIM_V2;

-- Create Maintenance type dimension
DROP TABLE MAINTENANCETYPE_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCETYPE_DIM_V2 AS
SELECT DISTINCT MAINTENANCETYPE
FROM MAINTENANCETYPE
ORDER BY MAINTENANCETYPE;

SELECT * FROM MAINTENANCETYPE_DIM_V2;

-- Create maintenance team dimension
DROP TABLE MAINTENANCETEAM_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCETEAM_DIM_V2 AS
SELECT DISTINCT M.TEAMID,
(1.0/COUNT(B.CENTERID)) AS RESEARCHCENTER_WEIGHTFACTOR,
LISTAGG(B.CENTERID, '_') WITHIN GROUP (ORDER BY B.CENTERID) AS
RESEARCHCENTER_LISTAGG
FROM MAINTENANCETEAM M, BELONGTO B
WHERE M.TEAMID = B.TEAMID
GROUP BY M.TEAMID
ORDER BY M.TEAMID;

SELECT * FROM MAINTENANCETEAM_DIM_V2;

-- Create maintenance team and research center bridge table
DROP TABLE TEAMCENTER_BRIDGE_V2 CASCADE CONSTRAINTS PURGE;
```

```
CREATE TABLE TEAMCENTER_BRIDGE_V2 AS
SELECT *
FROM BELONGTO
ORDER BY TEAMID;

SELECT * FROM TEAMCENTER_BRIDGE_V2;

-- Create research center dimension
DROP TABLE RESEARCHCENTER_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE RESEARCHCENTER_DIM_V2 AS
SELECT DISTINCT CENTERID
FROM RESEARCHCENTER
ORDER BY CENTERID;

SELECT * FROM RESEARCHCENTER_DIM_V2;

-- Create Maintenance Fact table for star schema v-2 (level 0)
DROP TABLE MAINTENANCE_FACT_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE MAINTENANCE_FACT_V2 AS
SELECT M.MAINTENANCEID, C.REGISTRATIONNO, M.MAINTENANCETYPE,
M.TEAMID,
SUM(M.MAINTENANCECOST) AS TOTAL_MAINTENANCE_COST,
COUNT(M.MAINTENANCEID) AS NO_OF_MAINTENANCE_RECORDS
FROM MAINTENANCE M, MAINTENANCETYPE MTY, MAINTENANCETEAM MTE, CAR C
WHERE M.MAINTENANCETYPE = MTY.MAINTENANCETYPE
AND M.TEAMID = MTE.TEAMID
AND M.REGISTRATIONNO = C.REGISTRATIONNO
GROUP BY M.MAINTENANCEID, C.REGISTRATIONNO, M.MAINTENANCETYPE,
M.TEAMID
ORDER BY M.MAINTENANCEID;

SELECT * FROM MAINTENANCE_FACT_V2;

-- Create error dimension
DROP TABLE ERRORCODE_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ERRORCODE_DIM_V2 AS
SELECT DISTINCT ERRORCODE
FROM ERROR
ORDER BY ERRORCODE;

SELECT * FROM ERRORCODE_DIM_V2;

-- Create accident zone dimension
DROP TABLE ACCIDENTZONE_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENTZONE_DIM_V2 AS
SELECT DISTINCT ACCIDENTZONE
FROM ACCIDENTINFO
ORDER BY ACCIDENTZONE;
```

```
SELECT * FROM ACCIDENTZONE_DIM_V2;

-- Create car damage severity dimension
DROP TABLE CARDAMAGESEVERITY_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE CARDAMAGESEVERITY_DIM_V2 AS
SELECT DISTINCT CAR_DAMAGE_SEVERITY
FROM ACCIDENTINFO
ORDER BY CAR_DAMAGE_SEVERITY;

SELECT * FROM CARDAMAGESEVERITY_DIM_V2;

-- Create accident info dimension
DROP TABLE ACCIDENTINFO_DIM_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENTINFO_DIM_V2 AS
SELECT DISTINCT A.ACCIDENTID,
(1.0/COUNT(C.REGISTRATIONNO)) AS CAR_WEIGHTFACTOR,
LISTAGG(C.REGISTRATIONNO, '_') WITHIN GROUP (ORDER BY C.REGISTRATIONNO)
AS CAR_LISTAGG
FROM ACCIDENTINFO A, CARACCIDENT C
WHERE A.ACCIDENTID = C.ACCIDENTID
GROUP BY A.ACCIDENTID
ORDER BY A.ACCIDENTID;

SELECT * FROM ACCIDENTINFO_DIM_V2;

-- Create accidentinfo and car bridge table
DROP TABLE ACCIDENTINFOCAR_BRIDGE_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENTINFOCAR_BRIDGE_V2 AS
SELECT *
FROM CARACCIDENT
ORDER BY ACCIDENTID;

SELECT * FROM ACCIDENTINFOCAR_BRIDGE_V2;

-- Create accident fact
DROP TABLE ACCIDENT_FACT_V2 CASCADE CONSTRAINTS PURGE;
CREATE TABLE ACCIDENT_FACT_V2 AS
SELECT DISTINCT A.ACCIDENTID, A.ACCIDENTZONE, A.ERRORCODE,
A.CAR_DAMAGE_SEVERITY,
COUNT(A.ACCIDENTID) AS NO_OF_ACCIDENT_RECORDS
FROM ACCIDENTINFO A, ERROR E
WHERE A.ERRORCODE = E.ERRORCODE
GROUP BY A.ACCIDENTID, A.ACCIDENTZONE, A.ERRORCODE,
A.CAR_DAMAGE_SEVERITY
ORDER BY A.ACCIDENTID;

SELECT * FROM ACCIDENT_FACT_V2;
```

3. OLAP Reports

3.1 OLAP Queries

3.1.1 Report 1

```
-- REPORT 1
SELECT BF.FACULTYID AS "FacultyID",
   BF.BOOKINGMONTH AS "Month",
   TO_CHAR(SUM(NO_OF_BOOKING_RECORDS), '9,999') AS "Total bookings",
   TO_CHAR(SUM(SUM(NO_OF_BOOKING_RECORDS)) OVER
   (ORDER BY TO_DATE(BF.BOOKINGMONTH, 'Month')
   ROWS UNBOUNDED PRECEDING),
   '9,999') AS "Cumulative number of booking records"
FROM BOOKING_FACT_V1 BF
WHERE BF.FACULTYID='FIT'
GROUP BY BF.FACULTYID, BF.BOOKINGMONTH
ORDER BY TO_DATE(BF.BOOKINGMONTH, 'Month');
```

	FacultyID	Month	Total bookings	Cumulative number of booking records
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

3.1.2. Report 2

```
SELECT
DECODE(GROUPING(MF.TEAMID), 1, 'All Teams', MF.TEAMID) AS "Team ID",
DECODE(GROUPING(MF.CARBODYTYPE), 1, 'All Car Body Types', MF.CARBODYTYPE)
AS "Car body type",
SUM(NO_OF_MAINTENANCE_RECORDS) AS "Total number of maintenance",
TO_CHAR(SUM(TOTAL_MAINTENANCE_COST),'9,999,999') AS "Total maintenance cost"
FROM MAINTENANCE_FACT_V1 MF
WHERE TEAMID='T002' OR TEAMID='T003'
GROUP BY CUBE(MF.TEAMID, MF.CARBODYTYPE);
```

	Team ID	Car body type	Total number of maintenance	Total maintenance cost
1	All Teams	All Car Body Types	399	125,300
2	All Teams	Bus	136	44,900
3	All Teams	Mini Bus	113	34,000
4	All Teams	People Mover	150	46,400
5	T002	All Car Body Types	197	62,700
6	T002	Bus	58	18,400
7	T002	Mini Bus	62	19,300
8	T002	People Mover	77	25,000
9	T003	All Car Body Types	202	62,600
10	T003	Bus	78	26,500
11	T003	Mini Bus	51	14,700
12	T003	People Mover	73	21,400

3.1.3. Report 3

```
-- REPORT 3
SELECT * FROM (
SELECT A.ERRORCODE AS "Error Code", A.REGISTRATIONNO AS "Registration No.",
A.CARBODYTYPE AS "Car Body Type", SUM(A.NO_OF_ACCIDENT_RECORDS) AS
"Total number of accidents",
DENSE_RANK() OVER (PARTITION BY A.ERRORCODE ORDER BY
SUM(A.NO_OF_ACCIDENT_RECORDS) DESC) AS "Rank"
FROM ACCIDENT_FACT_V1 A, CAR_DIM_V1 C
WHERE A.REGISTRATIONNO = C.REGISTRATIONNO
AND A.CARBODYTYPE = C.CARBODYTYPE
GROUP BY A.ERRORCODE, A.REGISTRATIONNO, A.CARBODYTYPE)
WHERE "Rank" <= 3;
```

	Error Code	Registration No.	Car Body Type	Total number of accidents	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
18	Error004	Car15	Mini Bus	10	2
19	Error004	Car18	Mini Bus	9	3
20	Error004	Car20	Mini Bus	9	3
21	Error004	Car04	Bus	9	3
22	Error005	Car16	Mini Bus	11	1
23	Error005	Car20	Mini Bus	10	2
24	Error005	Car08	Bus	10	2
25	Error005	Car05	Bus	9	3
26	Error005	Car19	Mini Bus	9	3
27	Error005	Car01	Bus	9	3
28	Error005	Car12	Mini Bus	9	3

3.1.4. Report 4

```
-- REPORT 4
SELECT B.CARBODYTYPE AS "Car body type",
DECODE (grouping (B.passengeragegroupid), 1, 'All Age Groups', B.passengeragegroupid)
AS "Age group",
DECODE (grouping (B.facultyid), 1, 'All Faculties', b.facultyid) AS "Faculty ID",
TO_CHAR(SUM(B.NO_OF_BOOKING_RECORDS),'9,999') AS "Total number of bookings"
FROM BOOKING_FACT_V1 B
WHERE B.CARBODYTYPE = 'People Mover'
GROUP BY B.CARBODYTYPE, CUBE(B.PASSENGERAGEGROUPID, B.FACULTYID);
```

	Car body type	Age group	Faculty ID	Total number of bookings
1	People Mover	All Age Groups	All Faculties	3,396
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	1,009
6	People Mover	All Age Groups	SCI	779
7	People Mover	Group1	All Faculties	1,380
8	People Mover	Group1	ART	169
9	People Mover	Group1	BUS	121
10	People Mover	Group1	ENG	382
11	People Mover	Group1	FIT	390
12	People Mover	Group1	SCI	318
13	People Mover	Group2	All Faculties	1,722
14	People Mover	Group2	ART	284
15	People Mover	Group2	BUS	193
16	People Mover	Group2	ENG	429
17	People Mover	Group2	FIT	497
18	People Mover	Group2	SCI	319
19	People Mover	Group3	All Faculties	294
20	People Mover	Group3	ENG	30
21	People Mover	Group3	FIT	122
22	People Mover	Group3	SCI	142

3.2. Reports with rollup and partial rollup

(a) Report 5: Using rollup with decode, how many booking records were made by each passenger age group (description) on each car body type on each month? Sort the result by booking month, passenger age group description and car body type.

Report 6: Modify Report 5 to use Partial Rollup (exclude “All Months” from the rollup).

(b) Rollup gets input from a set of attribute names to be grouped and produces subtotals of rolling-up aggregate combinations of the specified attributes and the grand total. Partial rollup is to include only some of the subtotals, no grand total aggregating across all attributes but only part of them.
In this case, rollup includes the sum of booking records for “All Months” and partial rollup doesn’t include. They only differ in one line of record.

3.2.1. Report 5

(c)

```
-- REPORT 5
SELECT BOOKINGMONTH AS "Booking month",
   PASSENGERAGEGROUPDESC AS "Passenger age group description",
   CARBODYTYPE AS "Car body type",
   SUM(NO_OF_BOOKING_RECORDS) AS "Number of Bookings"
```

```
FROM BOOKING_FACT_V1
GROUP BY ROLLUP (BOOKINGMONTH, PASSENGERAGEGROUPDESC,
CARBODYTYPE)
ORDER BY TO_DATE(BOOKINGMONTH, 'Month'), PASSENGERAGEGROUPDESC,
CARBODYTYPE;
```

(d)

	Booking month	Passenger age group description	Car body type	Number of Bookings
1	January	Middle-aged adults	Bus	134
2	January	Middle-aged adults	Mini Bus	154
3	January	Middle-aged adults	People Mover	142
4	January	Middle-aged adults	(null)	430
5	January	Old-aged adults	Bus	22
6	January	Old-aged adults	Mini Bus	9
7	January	Old-aged adults	People Mover	28
8	January	Old-aged adults	(null)	59
9	January	Young adults	Bus	135
10	January	Young adults	Mini Bus	130
11	January	Young adults	People Mover	130
12	January	Young adults	(null)	395
13	January	(null)	(null)	884
14	February	Middle-aged adults	Bus	159
15	February	Middle-aged adults	Mini Bus	104
16	February	Middle-aged adults	People Mover	134
17	February	Middle-aged adults	(null)	397
18	February	Old-aged adults	Bus	21
19	February	Old-aged adults	Mini Bus	17
20	February	Old-aged adults	People Mover	22
153	December	Young adults	Mini Bus	117
154	December	Young adults	People Mover	107
155	December	Young adults	(null)	339
156	December	(null)	(null)	847
157	(null)	(null)	(null)	10000

Note: Only the top 20 and bottom 5 rows are shown out of 157 rows in total

3.2.2 Report 6

(c)

```
-- REPORT 6
SELECT BOOKINGMONTH AS "Booking month",
   PASSENGERAGEGROUPDESC AS "Passenger age group description",
   CARBODYTYPE AS "Car body type",
   SUM(NO_OF_BOOKING_RECORDS) AS "Number of Bookings"
```

```
FROM BOOKING_FACT_V1
GROUP BY BOOKINGMONTH, ROLLUP(PASSENGERAGEGROUPDESC,
CARBODYTYPE)
ORDER BY TO_DATE(BOOKINGMONTH, 'Month'), PASSENGERAGEGROUPDESC,
CARBODYTYPE;
```

(d)

	Booking month	Passenger age group description	Car body type	Number of Bookings
1	January	Middle-aged adults	Bus	134
2	January	Middle-aged adults	Mini Bus	154
3	January	Middle-aged adults	People Mover	142
4	January	Middle-aged adults	(null)	430
5	January	Old-aged adults	Bus	22
6	January	Old-aged adults	Mini Bus	9
7	January	Old-aged adults	People Mover	28
8	January	Old-aged adults	(null)	59
9	January	Young adults	Bus	135
10	January	Young adults	Mini Bus	130
11	January	Young adults	People Mover	130
12	January	Young adults	(null)	395
13	January	(null)	(null)	884
14	February	Middle-aged adults	Bus	159
15	February	Middle-aged adults	Mini Bus	104
16	February	Middle-aged adults	People Mover	134
17	February	Middle-aged adults	(null)	397
18	February	Old-aged adults	Bus	21
19	February	Old-aged adults	Mini Bus	17
20	February	Old-aged adults	People Mover	22
152	December	Young adults	Bus	115
153	December	Young adults	Mini Bus	117
154	December	Young adults	People Mover	107
155	December	Young adults	(null)	339
156	December	(null)	(null)	847

Note: Only the top 20 and bottom 5 rows are shown out of 156 rows in total

3.3. Report with moving and cumulative aggregates

3.3.1. Report 7

(a) For each passenger age group, what is the moving average of booking records for the current month and preceding two months?

(b) This query is valuable for management to make observations in terms of trends of bookings, such as the amount of monthly bookings made for each age group, to be able to adapt and deploy suitable strategies such as promotions during suitable periods according to the moving average of booking records. For example, according to the report generated, old-aged adults have significantly less 3 month moving aggregate in April compared to young adults and middle-aged adults. With this information, the management can deploy marketing strategies such as promotion campaigns targeted at old-aged adults to attract more customers of the particular age group.
(c)

```
-- REPORT 7
SELECT BOOKINGMONTH AS "Booking Month", PASSENGERAGEGROUPDESC AS
"Passenger Age Group Description", SUM(NO_OF_BOOKING_RECORDS) AS "Number of
Booking Records",
ROUND(AVG(SUM(NO_OF_BOOKING_RECORDS)) OVER (ORDER BY
BOOKINGMONTH ROWS 2 PRECEDING),2) AS "Moving 3 Months Average"
FROM BOOKING_FACT_V1
GROUP BY BOOKINGMONTH, PASSENGERAGEGROUPDESC
ORDER BY TO_DATE(BOOKINGMONTH, 'Month');
```

(d)

	Booking Month	Passenger Age Group Description	Number of Booking Records	Moving 3 Months Average
1	January	Middle-aged adults	430	261.33
2	January	Old-aged adults	59	261
3	January	Young adults	395	294.67
4	February	Middle-aged adults	397	265
5	February	Young adults	294	250.33
6	February	Old-aged adults	60	265.33
7	March	Young adults	349	283.67
8	March	Old-aged adults	68	281.33
9	March	Middle-aged adults	434	284.33
10	April	Young adults	312	262.33
11	April	Old-aged adults	65	237.5
12	April	Middle-aged adults	410	410
13	May	Young adults	339	277.67
14	May	Old-aged adults	69	281
15	May	Middle-aged adults	425	280.67
16	June	Middle-aged adults	394	267.33

Note: Only the top 16 rows are shown out of 36 rows in total

3.3.2. Report 8

(a) What is the total maintenance cost for each car body type within each team and the cumulative maintenance cost for each team?
(b) This query is valuable for management to make observations such as which car body type costs the most in terms of maintenance cost in each team, and which maintenance team incurs the highest maintenance cost throughout all maintenance records. This is important as tracking maintenance cost is helpful in allowing the management to gain insight as to whether the type of maintenance work done is right or if regular maintenance has been done, since worse equipment damage means greater repair costs. The management can also decide on which management team to invest more as it is important to invest in the equipment to ensure adequate safety use and efficiency. For example, the management may

decide to invest more in Team 4 in terms of maintenance budget so more necessary and regular maintenance can be done.
(c)

```
-- REPORT 8
SELECT TEAMID AS "Team ID", CARBODYTYPE AS "Car Body Type",
SUM(TOTAL_MAINTENANCE_COST) AS "Total Maintenance Cost",
TO_CHAR(SUM(SUM(TOTAL_MAINTENANCE_COST)) OVER (PARTITION BY TEAMID
ORDER BY TEAMID,
CARBODYTYPE ROWS UNBOUNDED PRECEDING), '9,999,999.99') AS "Cumulative
Maintenance Cost"
FROM MAINTENANCE_FACT_V1
GROUP BY TEAMID, CARBODYTYPE;
```

(d)

	Team ID	Car Body Type	Total Maintenance Cost	Cumulative Maintenance Cost
1	T001	Bus	25900	25,900.00
2	T001	Mini Bus	18700	44,600.00
3	T001	People Mover	21600	66,200.00
4	T002	Bus	18400	18,400.00
5	T002	Mini Bus	19300	37,700.00
6	T002	People Mover	25000	62,700.00
7	T003	Bus	26500	26,500.00
8	T003	Mini Bus	14700	41,200.00
9	T003	People Mover	21400	62,600.00
10	T004	Bus	18900	18,900.00
11	T004	Mini Bus	13400	32,300.00
12	T004	People Mover	16100	48,400.00
13	T005	Bus	23800	23,800.00
14	T005	Mini Bus	22500	46,300.00
15	T005	People Mover	16500	62,800.00

4. Business Intelligence Report

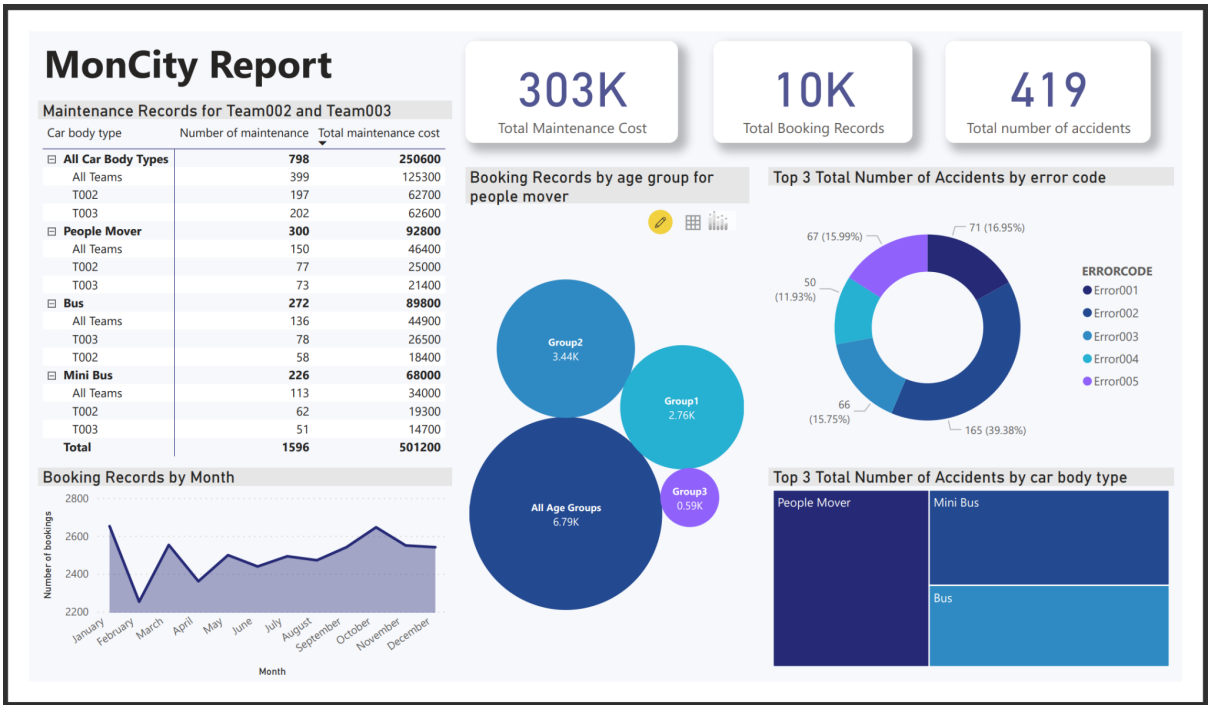

Figure 4. Screenshot of dashboard from OLAP Report 2,3,4 and 6

5. Final Recommendations/Suggestions

According to the area graph in the dashboard (Figure 4), we can observe that February has the least number of bookings compared to other months in the year. A suggestion to improve the current self-driving car project is that promotions can be made in February to attract more passengers.

Besides, the bubble chart in Figure 4 shows that the number of bookings made by old-aged adults is much lesser than the two other age groups. The reason for this might be some old-aged adults do not know how to make bookings especially in this smart city where almost everything is digitised. We suggest that the management can make the booking system more old-aged user friendly. Discounts can also be provided to old-aged adults to encourage them to place more bookings.

On the other hand, we can also observe from the donut chart in Figure 4 that Error code 002, which is low battery, contributed to the most accidents. To ensure the self-driving car project is safe for all passengers, accidents must be minimised. We suggest that the management can try to improve more on the self-driving car's battery life, make it more lasting and reduce the possibility of causing accidents due to low battery.

As for which car body type should be further invested, we suggest that mini bus should be a better choice. From the tables below (Figure 7), we are able to see that mini bus has 20 seats. It has more seats compared to people mover which can pick up more passengers at once. This can help reduce traffic congestion as it reduces the number of vehicles on the

road. In addition, we think that mini bus is a better option compared to bus in terms of number of seats because passengers might need to always wait for quite some time for the bus to be full before it departs.

In terms of number of bookings, referring to Figure 8, people mover has the most booking records but three of the car body types don't differ in a big manner. Hence, we think that it is equally worth it to invest in the three car body types when it comes to the number of bookings.

By analysing the accident records, we can see from Figure 9 that people mover is involved in the most accidents, mini bus is the second and bus is the least among three of them. In this case, we think that mini bus is safe enough as it has a relatively low number of accident records.

Finally, maintenance cost is also an important aspect to think about as if the self-driving car costs a lot when undergoing maintenance, it is a loss for the management. From Figure 10, mini bus has the lowest average maintenance cost compared to the other car body types. Therefore, by concluding all the different aspects discussed above, we think that mini bus is the car body type that is most worth it when it comes to further investing.

(500 words)

5.1. Supporting Data for Final Recommendations/Suggestions

```
SELECT A.ERRORCODE AS "Error Code", SUM(A.NO_OF_ACCIDENT_RECORDS) AS
"Total number of accidents"
FROM ACCIDENT_FACT_V1 A, CAR_DIM_V1 C
WHERE A.REGISTRATIONNO = C.REGISTRATIONNO
AND A.CARBODYTYPE = C.CARBODYTYPE
GROUP BY A.ERRORCODE
ORDER BY A.ERRORCODE;
```

	Error Code	Total number of accidents
1	Error001	169
2	Error002	373
3	Error003	163
4	Error004	145
5	Error005	150

Figure 5. The number of accidents for each error code

```
SELECT PASSENGERAGEGROUPDESC, SUM(NO_OF_BOOKING_RECORDS) AS
"Number of Bookings"
FROM BOOKING_FACT_V1
GROUP BY PASSENGERAGEGROUPDESC
ORDER BY PASSENGERAGEGROUPDESC;
```

	PASSENGERAGEGROUPDESC	Number of Bookings
1	Young adults	4085
2	Old-aged adults	853
3	Middle-aged adults	5062

Figure 6. The number of bookings based on each passenger age group

```
SELECT CARBODYTYPE, NUMSEATS
FROM CARBODYTYPE_DIM_V1
GROUP BY CARBODYTYPE, NUMSEATS
ORDER BY CARBODYTYPE;
```

	CARBODYTYPE	NUMSEATS
1	Bus	40
2	Mini Bus	20
3	People Mover	10

Figure 7. The number of seats for each car body type

```
SELECT C.CARBODYTYPE, COUNT(B.BOOKINGID) AS "Total number of bookings"
FROM BOOKING B, CAR C
WHERE B.REGISTRATIONNO=C.REGISTRATIONNO
GROUP BY C.CARBODYTYPE;
```

	CARBODYTYPE	Total number of bookings
1	Bus	3291
2	Mini Bus	3313
3	People Mover	3396

Figure 8. The number of booking records for each car body type

```
SELECT A.CARBODYTYPE AS "Car Body Type", SUM(A.NO_OF_ACCIDENT_RECORDS)
AS "Total number of accidents"
FROM ACCIDENT_FACT_V1 A, CAR_DIM_V1 C
WHERE A.REGISTRATIONNO = C.REGISTRATIONNO
AND A.CARBODYTYPE = C.CARBODYTYPE
GROUP BY A.CARBODYTYPE
ORDER BY A.CARBODYTYPE;
```

	Car Body Type	Total number of accidents
1	Bus	305
2	Mini Bus	324
3	People Mover	371

Figure 9. The number of accidents for each car body type

```
SELECT CARBODYTYPE, NO_OF_MAINTENANCE_RECORDS,
TOTAL_MAINTENANCE_COST,
```

```
ROUND(TOTAL_MAINTENANCE_COST/NO_OF_MAINTENANCE_RECORDS,2) AS
AVERAGE_MAINTENANCE_COST
FROM (
SELECT CARBODYTYPE,
   SUM(NO_OF_MAINTENANCE_RECORDS) AS NO_OF_MAINTENANCE_RECORDS,
   SUM(TOTAL_MAINTENANCE_COST) AS TOTAL_MAINTENANCE_COST
FROM MAINTENANCE_FACT_V1
GROUP BY CARBODYTYPE);
```

	CARBODYTYPE	NO_OF_MAINTENANCE_RECORDS	TOTAL_MAINTENANCE_COST	AVERAGE_MAINTENANCE_COST
1	Bus	355	113500	319.72
2	Mini Bus	311	88600	284.89
3	People Mover	334	100600	301.2

Figure 10. The average maintenance cost for each car body type

6. References

Bill The Lizard. (2009, Feb 9). *Removing duplicate rows from table in Oracle*. Stack Overflow. https://stackoverflow.com/questions/529098/removing-duplicate-rows-from-table-in-oracle