FIT3003 MonCity DataWarehouse

Ko Ko Win (31842305)
Mohamed Musthafa Mohamed Altaf (30864461)

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
31842305	Win	Ko Ko
30864461	Mohamed Altaf	Mohamed Musthafa

* Please include the names of all other group members.

Unit name and code	FIT3003	
Title of assignment	FIT3003 Major Assignment: MonCity	
Lecturer/tutor	Dr. Soon Lay Ki	
Tutorial day and time	Tuesday 8am - 10am	**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:** 11/10/2022

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 The material used can be from any source (staff, students or the internet, published and unpublished works).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 _____ Date: 11.10.2022 Signature _____ Date: _____

Signature _____ Date: 11.10.2022 Signature _____ Date: _____

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

[redacted]

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

[redacted]

Student ID	Student Name	Contribution Percentage
30864461	Mohamed Musthafa Mohamed Altaf - Task C.1 a ER diagram 6 entities. - Task C.1 b cleaned data for 6 entities. - Task C.1 c star schema version 1 - Task C.1 e explanation of the differences between the two star schema. - Task C.2 a - Task C.3 Report 3,4,7,8 - Task C.5 final recommendations	50%
31842305	Ko Ko Win - Task C.1 a ER diagram 6 entities. - Task C.1 b cleaned data for 6 entities. - Task C.1 c star schema version 2 - Task C.1 d explanation for the choice of dimensions - Task C.2 b - Task C.3 Report 1,2,5,6 - Task C.4 BI report	50%

[redacted]

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.

[redacted]

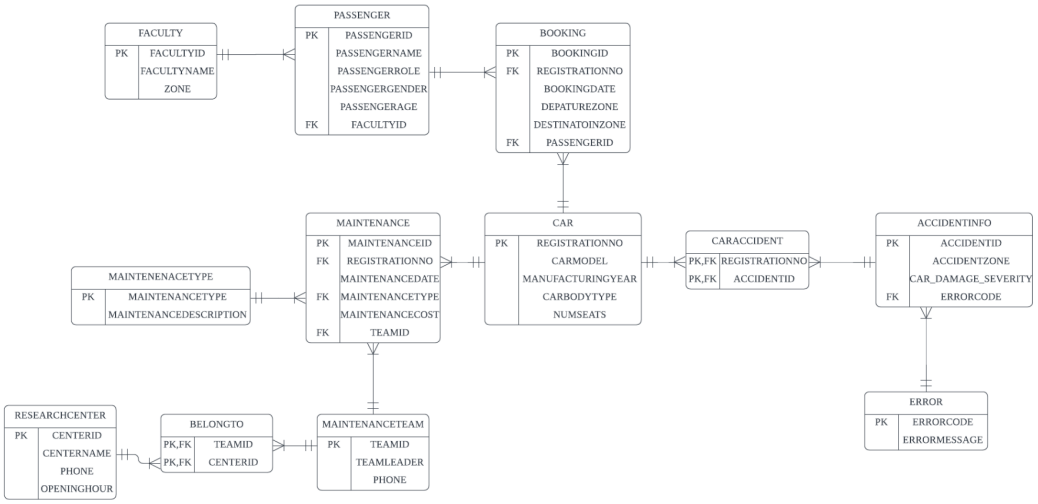
Signatures

Date 11/ 10 2022 Day Month Year

Table Of Contents

C.1. Data Warehouse Design

a) Operational Database ERD



b) Data Cleaning Strategies

For a relatively small table in the operational database, we used **SELECT * FROM TABLENAME;** to check the content of the table to manually check for any error. On the other hand, for a table with multiple rows, we use **“desc TableName”** to look for columns with not null constraints and check if any null values exist for that particular column.

To check for duplicate records in the table we used **Count(*)** to count the total number of rows in the table and **Count(distinct PK)** to count the total unique number of rows in the table. If the unique record is less than the total record in the table we will identify the duplicate record to remove it.

Additionally, if a table contains a foreign key we will check to ensure that a certain foreign key exists in the parent table by using the following command, **SELECT FK FROM TABLENAME WHERE FK NOT IN (SELECT FK FROM PARENTTBALENAME);**

Moreover, to check for null values in a column we use **SELECT * FROM TABLENAME WHERE PK IS NULL;** To delete a row that contains a null value from the table we use the following command **DELETE FROM TABLENAME WHERE COLUMNAME IS NULL;**

Other than that, when we have numerical data such as cost and sales we need to check if any insensible data exists. For instance, it is impossible for sales to be at a negative value. To check for insensible data we use SQL commands such as **SELECT * FROM TABLENAME WHERE COLUMN < 0;** If any negative values exist they will be removed.

List of Errors

1. **FACULTYID: Alienware** is an invalid foreign key in **PASSENGER** table as it does not exist in the parent table FACULTY.
2. Duplicate data value **BOOKINGID: T1218** in **BOOKING** table as it violates the unique property of the primary key.
3. **ERRORCODE: Error010** is an invalid foreign key in **ACCIDENTINFO** table as it does not exist in the parent table ERROR.
4. Null value exist in **ACCIDENTID** inside **ACCIDENTINFO** table**.**
5. One record for **MAINTENANCECOST** inside **MAINTENANCE** table has negative value. The cost of maintenance cannot be negative since it is logically impossible.

Error 1

FACULTYID: Alienware is an invalid foreign key which does not exist in the parent table(FACULTY)

Detect Error

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

Output

	PASSENGERID	PASSENGERNAME	PASSENGERROLE	PASSENGERGENDER	PASSENGERAGE	FACULTYID
1	U163	Anabia Mccabe	Staff	Male	21	Alienware
2	U010	Jeremy Chandler	Student	Female	25	ENG
3	U011	Marquis Lozano	Staff	Male	55	ENG
4	U012	Scarlett Berry	Staff	Male	31	ENG
5	U013	Salvador Salinas	Staff	Male	68	ENG

Fix Error

```
CREATE TABLE passenger AS
SELECT * FROM moncity.passenger
WHERE facultyid IN (SELECT facultyid FROM moncity.faculty);
```

Output

	PASSENGERID	PASSENGERNAME	PASSENGERROLE	PASSENGERGENDER	PASSENGERAGE	FACULTYID
1	U118	Abbey Key	Staff	Male	50	SCI
2	U145	Aidan Solomon	Student	Male	45	FIT
3	U109	Alannah Wilcox	Student	Female	39	FIT
4	U079	Alberto Rivers	Student	Female	18	ART
5	U117	Alexia Simon	Staff	Male	61	SCI

Error 2

Duplicate BOOKINGID exist in BOOKING table.

Detect Error

```
-- Check for duplicate data
SELECT Count(*) FROM  moncity.booking; -- 10001

SELECT Count(DISTINCT bookingid) FROM  moncity.booking; -- 10000
```

Output

	BOOKINGID	REGISTRATIO...	BOOKINGDATE	DEPARTUREZONE	DESTINATIONZONE	PASSENGERID
407	T1218	Car14	29/07/2018	ZoneD	ZoneB	U059
408	T1218	Car14	29/07/2018	ZoneD	ZoneB	U059
409	T1219	Car27	29/07/2018	ZoneA	ZoneA	U160
410	T122	Car04	25/11/2015	ZoneC	ZoneA	U099
411	T1220	Car29	17/03/2021	ZoneC	ZoneC	U058

Fix Error

```
CREATE TABLE booking AS
SELECT DISTINCT * FROM moncity.booking;
```

Output

	BOOKINGID	REGISTRATIONNO	BOOKINGDATE	DEPARTUREZONE	DESTINATIONZONE	PASSENGERID
407	T1218	Car14	29/07/2018	ZoneD	ZoneB	U059
408	T1219	Car27	29/07/2018	ZoneA	ZoneA	U160
409	T122	Car04	25/11/2015	ZoneC	ZoneA	U099
410	T1220	Car29	17/03/2021	ZoneC	ZoneC	U058

Error 3

ERRORCODE: Error010 is an invalid foreign key which does not exist in the parent table (ERROR).

Detect Error

```
SELECT *
FROM moncity.accidentinfo
WHERE errorcode NOT IN (SELECT errorcode FROM moncity.error);
```

Output

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
1	A2000	ZoneB	No damage	Error010

Fix Error

```
CREATE TABLE accidentinfo AS
SELECT *
FROM moncity.accidentinfo WHERE errorcode  IN (SELECT errorcode FROM
moncity.error);
```

Output

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
1	A632	ZoneA	No damage	Error002
2	A633	ZoneB	No damage	Error002
3	A634	ZoneB	Severe damage	Error001
4	A635	ZoneD	Severe damage	Error001

Error 4

Null value exist in ACCIDENTID inside the ACCIDENTINFO table.

Detect Error

```
---Check for Null values
SELECT * FROM accidentinfo WHERE accidentid IS NULL;
```

Output

	ACCID...	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
1	(null)	ZoneC	Severe damage	Error005
2	A999	ZoneC	Very minor damage	Error005
3	A998	ZoneA	Severe damage	Error001
4	A997	ZoneA	Very minor damage	Error001
5	A996	ZoneC	Minor damage	Error004
6	A995	ZoneC	Very minor damage	Error004

Fix Error

```
DELETE FROM accidentinfo
WHERE accidentid IS NULL;
```

Output

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
1	A999	ZoneC	Very minor damage	Error005
2	A998	ZoneA	Severe damage	Error001
3	A997	ZoneA	Very minor damage	Error001
4	A996	ZoneC	Minor damage	Error004
5	A995	ZoneC	Very minor damage	Error004

Error 5

Negative value for maintenance cost in maintenance table

Detect Error

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

Output

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

Fixing the error

```
-- FIXING THE ERROR
```

```sql
DROP TABLE maintenance;
CREATE TABLE maintenance AS
SELECT *
FROM moncity.maintenance
WHERE maintenancecost > 0;

select *
from maintenance
where maintenancecost < 0;
```

Output

MAINTENANCEID	REGISTRATIONNO	MAINTENANCEDATE	MAINTENANCETYPE	MAINTENANCECOST	TEAMID

Full SQL commands for Data Cleaning

```sql
/*
Members: Ko Ko Win, Muhammad Musthafa Althaf
ERRORS:
1) invalid foreign key (facultyid) exist in moncity.passenger
2) duplicates record in moncity.booking
3)  invalid foreign key (errorcode) exist in moncity.passenger
4) null value in moncity.caraccident primary key (accidentid)
5) 1 record has negative value for maintenancecost in
moncity.maintenancecost
*/

-- drop tables
DROP TABLE faculty CASCADE CONSTRAINTS;
DROP TABLE researchcenter CASCADE CONSTRAINTS;
DROP TABLE passenger CASCADE CONSTRAINTS;
DROP TABLE error CASCADE CONSTRAINTS;
DROP TABLE maintenancetype CASCADE CONSTRAINTS;
DROP TABLE booking CASCADE CONSTRAINTS;
DROP TABLE accidentinfo CASCADE CONSTRAINTS;
DROP TABLE car  CASCADE CONSTRAINTS;
DROP TABLE maintenance CASCADE CONSTRAINTS;
DROP TABLE caraccident CASCADE CONSTRAINTS;
DROP TABLE maintenanceteam CASCADE CONSTRAINTS;
DROP TABLE belongto  CASCADE CONSTRAINTS;
```

```sql
-- Creating local tables
----------------FACULTY (No Error) ------------------
SELECT
    *
FROM
    moncity.faculty;

CREATE TABLE faculty AS
SELECT * FROM moncity.faculty;


-------------------RESEARCHCENTER (No Error) ------------
SELECT
    *
FROM
    moncity.researchcenter;

CREATE TABLE researchcenter AS
SELECT * FROM moncity.researchcenter;

------------------------PASSENGER (1 Error)------------------------
/*
Invalid facultyid in row 143
*/
desc moncity.passenger;
select * from moncity.passenger;
-- check for null values
SELECT DISTINCT
    *
FROM
    moncity.passenger
WHERE
    PASSENGERID IS NULL OR FACULTYID IS NULL;
-- check for duplicate values
SELECT PASSENGERID,FACULTYID,count(*) from moncity.passenger
GROUP BY
    PASSENGERID,FACULTYID
HAVING
    COUNT(*) > 1;

-- detect error
```

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

-- create table and fix error
CREATE TABLE passenger AS
SELECT * FROM moncity.passenger
WHERE facultyid IN (SELECT facultyid FROM moncity.faculty);

select * from passenger;

------------------------ERROR (No Error)---------------------
desc moncity.error;

CREATE TABLE error AS
SELECT * FROM moncity.error;

------------------------MAINTENANCETYPE (No Error)-------------
desc moncity.maintenancetype;

select * from moncity.maintenancetype;

CREATE TABLE maintenancetype AS
SELECT * FROM moncity.maintenancetype;


--------------------------BOOKING (1 Error)-----------------------
desc moncity.booking;

SELECT * FROM moncity.booking;

-- Check for duplicate data
SELECT Count(*) FROM  moncity.booking; -- 10001

SELECT Count(DISTINCT bookingid) FROM  moncity.booking; -- 10000

-- FIND DUPLICATE VALUE (T1218)
SELECT bookingid,registrationno,count(*) from moncity.booking
GROUP BY
     bookingid,registrationno
HAVING
    COUNT(*) > 1;
```

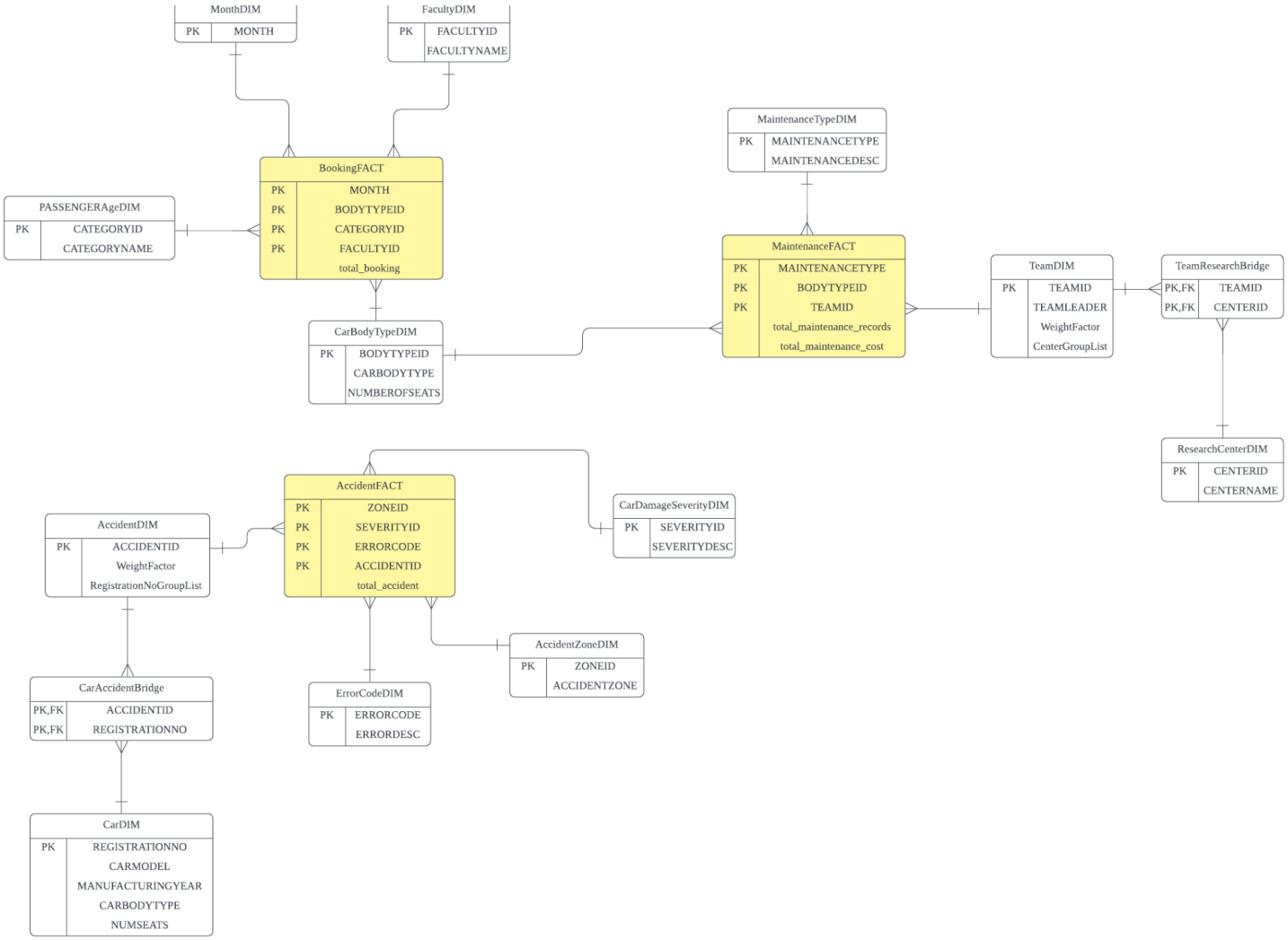
```sql
-- CHECK THE DUPLICATE RECORD
SELECT * FROM moncity.booking WHERE bookingid = 'T1218';

--FIX ERROR
CREATE TABLE booking AS
SELECT DISTINCT * FROM moncity.booking;

SELECT * FROM booking;

------------------------ACCIDENTINFO (2 Errors)----------------
/* Invalid  foreign key error code (Error010)  exist which was not
present in parent table*/
select * from moncity.accidentinfo;

-- ERROR DETECT
SELECT *
FROM moncity.accidentinfo
WHERE errorcode NOT IN (SELECT errorcode FROM moncity.error);

-- FIX ERROR
CREATE TABLE accidentinfo AS
SELECT *
FROM moncity.accidentinfo WHERE errorcode  IN (SELECT errorcode FROM
moncity.error);


---Check for Null values
SELECT * FROM accidentinfo WHERE accidentid IS NULL;

--Fix error
DELETE FROM accidentinfo
WHERE accidentid IS NULL;


SELECT * FROM accidentinfo WHERE accidentid IS NULL;

SELECT * FROM accidentinfo;


------------------------CAR (No Error)----------------
--CHECKING FOR ERRORS
SELECT * FROM moncity.car;
```

```sql
--CREATE NEW TABLE
CREATE TABLE car AS
SELECT *
FROM moncity.car;

select * from car;

------------------------MAINTENANCE (1 Error)----------------


-- FINDING ERROR
SELECT *
FROM moncity.maintenance
WHERE maintenancecost <= 0;

-- FIXING THE ERROR
CREATE TABLE maintenance AS
SELECT *
FROM moncity.maintenance
WHERE maintenancecost > 0;

SELECT * FROM maintenance;


------------------------CARACCIDENT (No Error)----------------

--CHECKING FOR ERRORS
SELECT *
FROM
(SELECT accidentid, registrationno, count(*) AS NUM_OF_UNIQUEROWS
FROM moncity.caraccident
GROUP BY accidentid, registrationno)
WHERE NUM_OF_UNIQUEROWS > 1 OR accidentid IS NULL OR registrationno IS
NULL;

--NO NULL VALUES OR DUPLICATE VALUES

--CREATE TABLE
CREATE TABLE caraccident AS
SELECT *
FROM moncity.caraccident;

SELECT * FROM caraccident;
```

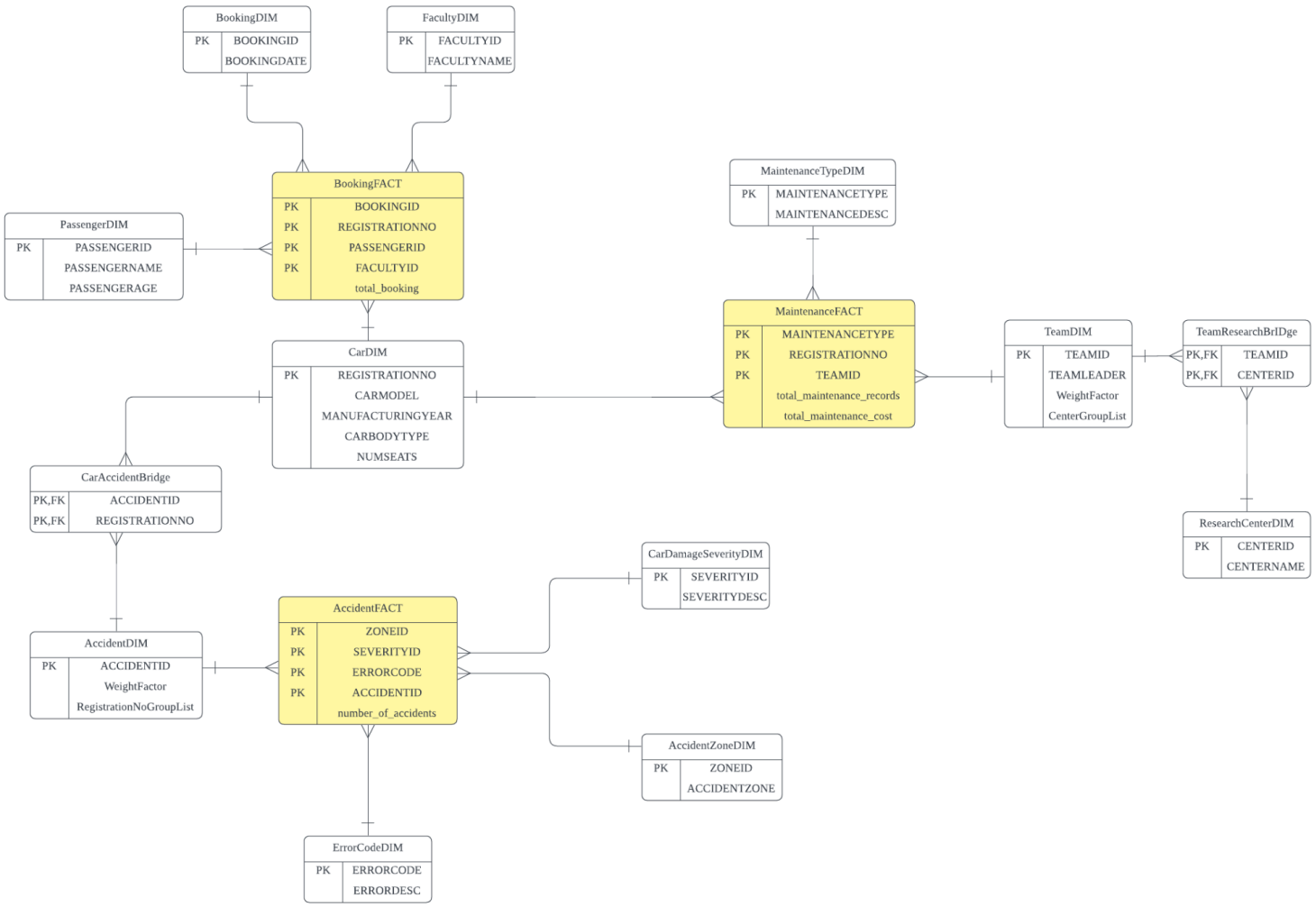
```sql
-------------------------MAINTENANCETEAM (No Error)-----------------
--no errors
SELECT * FROM moncity.maintenanceteam;

-- CREATE TABLE
CREATE TABLE maintenanceteam AS
SELECT *
FROM moncity.maintenanceteam;

SELECT * FROM maintenanceteam;


------------------------BELONGTO (No Error)-----------------
--no errors
SELECT * FROM moncity.belongto;

-- CREATE TABLE
CREATE TABLE belongto AS
SELECT *
FROM moncity.belongto;

SELECT * FROM belongto;

select * from moncity.accidentinfo;
```

c) Star Schema Design

Star Schema Level 2 (Version-1)



Star Schema Level 0 (Version-2)



d) Reasoning For Choice Of Dimensions

Determinant Dimensions

The determinant dimension is not present in our star schema. After further clarification, instead of using the car dimension as a determinant dimension to calculate the total number of accidents, we have taken an alternative design approach by using a bridge table between the accident dimension and the car dimension. Therefore, we do not need to worry about double counting and accuracy in retrieving the car registration number.

Slowly Changing Dimensions type

We did not implement any SCD type for any of our dimensions. It is because there are no attributes present that has a lifetime and there is no method to track changes in the operational database.

e) Differences Between The Two Star Schema

Differences between two star schema

In the level 0 star schema, the following changes were made to our level 2 star schema:

- **Booking Dimension** replaces **Month Dimension**.
- **Passenger Dimension** replaces **Age Category Dimension**.

The above changes introduce PassengerID and Passenger age for each passenger instead of the passenger age category. Moreover, the Booking ID and Booking date for each booking were introduced replacing the Month dimension for booking. These changes eliminate all the aggregation in the fact measure of the Booking and Passenger fact table by increasing the granularity.

Moreover, In our level 2 star schema CarBodyType dimension is connected to the Booking fact table and Maintenance fact table. Whereas, In our level-0 star schema the following changes were made.

- **Car Dimension** replaces the **CarBodyType Dimension.**

The above changes ensure that the data in the dimension become more detailed which will eliminate aggregation of the fact measure in the Booking fact table and Maintenance fact table by introducing the Registration Number of each car.

In conclusion, inside our version 2 star schema we have lowered the level of aggregation of the version 1 star schema by changing the granularity of the existing dimensions with higher granularity dimensions. Thus, in our version 2 star schema the value of fact measures will be broken down into more detail which reaches to Level-0 star schema with the lowest level of aggregation.

C.2. Star Schema Implementation

Note:

- *"MonCity_fact_2" = Version-1 (Level 2)*
- *"MonCity_fact_0" = Version-2 (Level 0)*

a) Star Schema Level 2 Implementation (Version-1)

Drop Tables Commands

```
-- Drop dimension & fact tables
DROP TABLE faculty_dim_2;

DROP TABLE month_dim_2;

DROP TABLE passengerage_dim_2;

DROP TABLE carbodytype_dim_2;

DROP TABLE booking_tempfact_2;

DROP TABLE booking_fact_2;

DROP TABLE errorcode_dim_2;

DROP TABLE accidentzone_dim_2;

DROP TABLE cardamageseverity_dim_2;

DROP TABLE car_dim_2;

DROP TABLE car_accident_bridge_2;

DROP TABLE accident_dim_2;

DROP TABLE accident_fact_2;

DROP TABLE accident_temp_fact_2;

DROP TABLE maintenancetype_dim_2;

DROP TABLE team_dim_2;
```

```
DROP TABLE teamresearch_bridge_2;

DROP TABLE researchcenter_dim_2;

DROP TABLE maintenance_temp_fact_2;
DROP TABLE maintenance_fact_2;
```

Faculty Dimension

```
-- FacultyDIM
CREATE TABLE faculty_dim_2 AS
SELECT FACULTYID, FACULTYNAME
FROM FACULTY;
```

	FACULTYID	FACULTYNAME
1	FIT	Information Technology
2	BUS	Business and Economics
3	ENG	Engineering
4	ART	Art, Design and Architecture
5	SCI	Science

Month Dimension

```
-- MonthDIM
CREATE TABLE month_dim_2 AS
SELECT DISTINCT to_char(BOOKINGDATE, 'MM') AS MONTH
FROM BOOKING
ORDER BY MONTH asc;
```

	MONTH
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
10	10
11	11
12	12

PassengerAge Dimension

```
-- PassengerAgeDIM
CREATE TABLE passengerage_dim_2 (CATEGORYID NUMBER , CATEGORYNAME
VARCHAR(20));

INSERT INTO passengerage_dim_2 VALUES (1, 'Young Adults');
INSERT INTO passengerage_dim_2 VALUES (2, 'Middle Aged Adults');
INSERT INTO passengerage_dim_2 VALUES (3, 'Old Aged Adults');
```

	CATEGORYID	CATEGORYNAME
1	1	Young Adults
2	2	Middle Aged Adults
3	3	Old Aged Adults

CarBodyType Dimension

```
-- CarBodyTypeDIM
CREATE TABLE carbodytype_dim_2 AS
SELECT DISTINCT CARBODYTYPE, NUMSEATS
FROM CAR;

ALTER TABLE carbodytype_dim_2
ADD (BODYTYPEID NUMBER);

UPDATE carbodytype_dim_2
SET BODYTYPEID = 1
```

```
WHERE CARBODYTYPE = 'Bus';

UPDATE carbodytype_dim_2
SET BODYTYPEID = 2
WHERE CARBODYTYPE = 'Mini Bus';

UPDATE carbodytype_dim_2
SET BODYTYPEID = 3
WHERE CARBODYTYPE = 'People Mover';
```

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

Booking Temp Fact Table

```
------ Booking Fact Level 2

-- Booking Temp Fact
CREATE TABLE booking_tempfact_2 AS
SELECT p.PASSENGERID ,
p.PASSENGERAGE,
b.BOOKINGID,
to_char(b.BOOKINGDATE, 'MM') AS MONTH,
f.FACULTYID ,
c.REGISTRATIONNO ,
c.CARBODYTYPE
FROM PASSENGER p, FACULTY f, BOOKING b, CAR c
WHERE p.facultyid = f.facultyid
AND b.passengerid = p.passengerid
AND b.registrationno = c.registrationno ;

ALTER TABLE booking_tempfact_2
ADD (CATEGORYID NUMBER);

UPDATE booking_tempfact_2
SET CATEGORYID = 1
WHERE PASSENGERAGE >= 18 AND PASSENGERAGE <= 35;

UPDATE booking_tempfact_2
```

```
SET CATEGORYID = 2
WHERE PASSENGERAGE >= 36 AND PASSENGERAGE <= 59;

UPDATE booking_tempfact_2
SET CATEGORYID = 3
WHERE PASSENGERAGE >= 60;

ALTER TABLE booking_tempfact_2
ADD(BODYTYPEID NUMBER);

UPDATE booking_tempfact_2
SET BODYTYPEID = 1
WHERE CARBODYTYPE = 'Bus';

UPDATE booking_tempfact_2
SET BODYTYPEID = 2
WHERE CARBODYTYPE = 'Mini Bus';

UPDATE booking_tempfact_2
SET BODYTYPEID = 3
WHERE CARBODYTYPE = 'People Mover';
```

	PASSENGERID	PASSENGERAGE	BOOKINGID	MONTH	FACULTYID	REGISTRATIONNO	CARBODYTYPE	CATEGORYID	BODYTYPEID
1	U048	34	T716	01	FIT	Car06	Bus	1	1
2	U105	57	T717	09	BUS	Car09	Bus	2	1
3	U098	34	T726	11	ART	Car26	People Mover	1	3
4	U126	43	T730	05	SCI	Car06	Bus	2	1
5	U114	67	T740	10	FIT	Car12	Mini Bus	3	2
6	U132	24	T758	01	FIT	Car17	Mini Bus	1	2
7	U056	41	T789	02	ART	Car19	Mini Bus	2	2
8	U159	24	T797	09	BUS	Car23	People Mover	1	3
9	U125	22	T819	01	SCI	Car02	Bus	1	1
10	U120	25	T822	03	ENG	Car22	People Mover	1	3
11	U016	47	T823	04	ENG	Car20	Mini Bus	2	2
12	U112	47	T828	07	FIT	Car01	Bus	2	1

Booking Fact Table

```
-- BookingFact
CREATE TABLE booking_fact_2 AS
SELECT
MONTH,
BODYTYPEID,
CATEGORYID,
FACULTYID,
Count(*) AS total_booking
FROM booking_tempfact_2
GROUP BY
```

```
MONTH,
BODYTYPEID,
CATEGORYID,
FACULTYID
ORDER BY MONTH ASC;
```

	MONTH	BODYTYPEID	CATEGORYID	FACULTYID	TOTAL_BOOKING
1	01	1	1	ART	19
2	01	1	1	BUS	12
3	01	1	1	ENG	38
4	01	1	1	FIT	38
5	01	1	1	SCI	28
6	01	1	2	ART	19
7	01	1	2	BUS	17
8	01	1	2	ENG	35
9	01	1	2	FIT	40
10	01	1	2	SCI	23
11	01	1	3	ENG	2
12	01	1	3	FIT	10

ErrorCode Dimension

```
-- ErrorCodeDIM
CREATE TABLE errorcode_dim_2 AS
SELECT * FROM error;
```

	ERRORCODE	ERRORMESSAGE
1	Error001	Image recognition system failed
2	Error002	Low Battery
3	Error003	Signal loss: front sensor
4	Error004	Lidar systeem failed:Unable to locate
5	Error005	Unknow issue: please contact with maintenance team

Accident Zone Dimension

```
-- AccidentZoneDIM
CREATE TABLE accidentzone_dim_2(ZONEID NUMBER, ACCIDENTZONE
VARCHAR(20));

INSERT INTO accidentzone_dim_2 VALUES(1, 'ZoneA');
INSERT INTO accidentzone_dim_2 VALUES(2, 'ZoneB');
INSERT INTO accidentzone_dim_2 VALUES(3, 'ZoneC');
INSERT INTO accidentzone_dim_2 VALUES(4, 'ZoneD');
```

	ZONEID	ACCIDENTZONE
1	1	ZoneA
2	2	ZoneB
3	3	ZoneC
4	4	ZoneD

CarDamageSeverity Dimension

```
-- CarDamageSeverityDIM
CREATE TABLE cardamageseverity_dim_2(SEVERITYID NUMBER, SEVERITYDESC
VARCHAR(20));

INSERT INTO cardamageseverity_dim_2 VALUES(1, 'No damage');
INSERT INTO cardamageseverity_dim_2 VALUES(2, 'Very minor damage');
INSERT INTO cardamageseverity_dim_2 VALUES(3, 'Minor damage');
INSERT INTO cardamageseverity_dim_2 VALUES(4, 'Moderate damage');
INSERT INTO cardamageseverity_dim_2 VALUES(5, 'Severe damage');
```

	SEVERITYID	SEVERITYDESC
1	1	No damage
2	2	Very minor damage
3	3	Minor damage
4	4	Moderate damage
5	5	Severe damage

Car Dimension

```
-- CarDIM
CREATE TABLE car_dim_2 AS SELECT * FROM CAR;
```

	REGISTRATIONNO	CARMODEL	MANUFACTURINGYEAR	CARBODYTYPE	NUMSEATS
1	Car01	CapitalX	2010	Bus	40
2	Car02	CapitalX pro	2010	Bus	40
3	Car03	CapitalY	2011	Bus	40
4	Car04	CapitalY pro	2011	Bus	40
5	Car05	RunnerV1	2006	Bus	40
6	Car06	RunnerV2	2007	Bus	40
7	Car07	RunnerV3	2008	Bus	40
8	Car08	RunnerV4	2009	Bus	40
9	Car09	RunnerV5	2010	Bus	40
10	Car10	RunnerV6	2011	Bus	40
11	Car11	TrailblazerA	2012	Mini Bus	20
12	Car12	TrailblazerA pro	2012	Mini Bus	20

CarAccident Bridge Table

```
-- CarAccidentBridge
CREATE TABLE car_accident_bridge_2 AS SELECT * FROM CARACCIDENT;
```

	REGISTRATIONNO	ACCIDENTID
1	Car06	A308
2	Car14	A309
3	Car04	A310
4	Car01	A311
5	Car24	A312
6	Car21	A313
7	Car24	A314
8	Car24	A315
9	Car19	A316
10	Car26	A317
11	Car02	A318
12	Car24	A319

Accident Dimension

```
-- AccidentDIM
CREATE TABLE accident_dim_2 AS
SELECT A.ACCIDENTID,
ROUND(1.0/COUNT(CA.REGISTRATIONNO),2) WeightFactor,
LISTAGG(CA.REGISTRATIONNO, '_') WITHIN GROUP(ORDER BY
CA.REGISTRATIONNO) AS RegistrationNoGroupList
FROM ACCIDENTINFO A, CARACCIDENT CA
WHERE A.ACCIDENTID = CA.ACCIDENTID
GROUP BY A.ACCIDENTID;
```

	ACCIDENTID	WEIGHTFACTOR	REGISTRATIONNOGROUPLIST
1	A1000	1	Car13
2	A1001	1	Car13
3	A1002	1	Car05
4	A1003	1	Car25
5	A1004	1	Car24
6	A1005	1	Car08
7	A1006	1	Car23
8	A1007	1	Car08
9	A1008	1	Car16
10	A1009	1	Car09
11	A1010	1	Car11
12	A1011	1	Car23

Accident Temp Fact Table

```
-- ACCIDENT Temp FACT TABLE
CREATE TABLE accident_temp_fact_2 AS
SELECT
A.ACCIDENTID,
A.ACCIDENTZONE,
A.CAR_DAMAGE_SEVERITY,
E.ERRORCODE,
COUNT(A.ACCIDENTID) AS TOTAL_ACCIDENT
FROM ACCIDENTINFO A, ERROR E
WHERE A.ERRORCODE = E.ERRORCODE
GROUP BY A.ACCIDENTID, A.ACCIDENTZONE, A.CAR_DAMAGE_SEVERITY,
E.ERRORCODE;

ALTER TABLE accident_temp_fact_2
    ADD(ZONEID NUMBER);

UPDATE accident_temp_fact_2
SET ZONEID = 1
WHERE ACCIDENTZONE = 'ZoneA' ;

UPDATE accident_temp_fact_2
SET ZONEID = 2
WHERE ACCIDENTZONE = 'ZoneB' ;

UPDATE accident_temp_fact_2
SET ZONEID = 3
WHERE ACCIDENTZONE = 'ZoneC' ;

UPDATE accident_temp_fact_2
SET ZONEID = 4
WHERE ACCIDENTZONE = 'ZoneD' ;

ALTER TABLE accident_temp_fact_2
    ADD(SEVERITYID NUMBER);

UPDATE accident_temp_fact_2
SET SEVERITYID = 1
WHERE CAR_DAMAGE_SEVERITY = 'No damage';

UPDATE accident_temp_fact_2
SET SEVERITYID = 2
WHERE CAR_DAMAGE_SEVERITY = 'Very minor damage';
```

```
UPDATE accident_temp_fact_2
SET SEVERITYID = 3
WHERE CAR_DAMAGE_SEVERITY = 'Minor damage';

UPDATE accident_temp_fact_2
SET SEVERITYID = 4
WHERE CAR_DAMAGE_SEVERITY = 'Moderate damage';

UPDATE accident_temp_fact_2
SET SEVERITYID = 5
WHERE CAR_DAMAGE_SEVERITY = 'Severe damage';
```

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE	TOTAL_ACCIDENT	ZONEID	SEVERITYID
1	A647	ZoneD	No damage	Error002	1	4	1
2	A648	ZoneB	No damage	Error002	1	2	1
3	A665	ZoneA	Severe damage	Error004	1	1	5
4	A669	ZoneD	No damage	Error002	1	4	1
5	A695	ZoneB	Minor damage	Error003	1	2	3
6	A699	ZoneA	No damage	Error002	1	1	1
7	A701	ZoneD	No damage	Error002	1	4	1
8	A719	ZoneA	No damage	Error002	1	1	1
9	A722	ZoneC	Very minor damage	Error005	1	3	2
10	A737	ZoneA	Very minor damage	Error004	1	1	2
11	A742	ZoneD	Minor damage	Error005	1	4	3
12	A747	ZoneA	Severe damage	Error005	1	1	5

Accident Fact Table

```
--AccidentFact
CREATE TABLE accident_fact_2 AS
SELECT ZONEID ,
SEVERITYID,
ERRORCODE,
ACCIDENTID,
COUNT(*) AS TOTAL_ACCIDENT
FROM accident_temp_fact_2
GROUP BY ZONEID,
SEVERITYID,
ERRORCODE,
ACCIDENTID;
```

	ZONEID	SEVERITYID	ERRORCODE	ACCIDENTID	TOTAL_ACCIDENT
1	1	5	Error005	A747	1
2	1	4	Error003	A765	1
3	4	4	Error003	A771	1
4	2	4	Error001	A316	1
5	2	2	Error005	A611	1
6	4	5	Error004	A1092	1
7	1	1	Error002	A1161	1
8	4	1	Error002	A851	1
9	3	5	Error001	A903	1
10	1	1	Error002	A916	1
11	3	5	Error005	A721	1
12	2	3	Error003	A423	1

MaintenanceType Dimension

```
-- MaintenanceTypeDIM
CREATE TABLE maintenancetype_dim_2 AS
SELECT *
FROM MAINTENANCETYPE;
```

	MAINTENANCETYPE	MAINTENANCEDESCRIPTION
1	M001	System Upgrade
2	M002	Hardware Upgrade:Sensor
3	M003	Hardware Upgrade:Lidar system
4	M004	Battery replacement
5	M005	Regular maintenance

ResearchCenter Dimension

```
-- ResearchCenterDIM
CREATE TABLE researchcenter_dim_2 AS
SELECT CENTERID, CENTERNAME
FROM RESEARCHCENTER;
```

	CENTERID	CENTERNAME
1	CE01	Skunk Works
2	CE02	Boeing Phantom Works
3	CE03	SRI International
4	CE04	Palo Alto Research Center Incorporated

TeamResearch Bridge Table

```
-- TeamResearch Bridge table
CREATE TABLE teamresearch_bridge_2 AS
SELECT *
FROM BELONGTO;
```

	TEAMID	CENTERID
1	T001	CE01
2	T001	CE02
3	T001	CE03
4	T001	CE04
5	T002	CE03
6	T002	CE04
7	T003	CE02
8	T004	CE01
9	T005	CE01
10	T005	CE02
11	T005	CE03
12	T005	CE04

Team Dimension

```
-- TeamDIM
CREATE TABLE team_dim_2 AS
SELECT m.TEAMID, m.TEAMLEADER, 1/count(*) AS WeightFactor,
LISTAGG(b.CENTERID, '_') Within Group (Order By b.CENTERID) AS
CenterGroupList
FROM maintenanceteam m, belongto b
WHERE m.TEAMID = b.TEAMID
GROUP BY m.TEAMID, m.TEAMLEADER;
```

	TEAMID	TEAMLEADER	WEIGHTFACTOR	CENTERGROUPLIST
1	T001	Guillermo Nash	0.25	CE01_CE02_CE03_CE04
2	T002	Adriel Gates	0.5	CE03_CE04
3	T003	Lillian Krueger	1	CE02
4	T004	Essence Bass	1	CE01
5	T005	Brady Mcconnell	0.25	CE01_CE02_CE03_CE04

Maintenance Temp Fact Table

```
--MaintenanceTempFact
CREATE TABLE maintenance_temp_fact_2 AS
SELECT mt.MAINTENANCETYPE, t.TEAMID, m.REGISTRATIONNO, c.CARBODYTYPE,
m.MAINTENANCEID , m.MAINTENANCECOST
FROM maintenancetype_dim_2 mt, maintenance m, team_dim_2 t, CAR c
WHERE m.TEAMID = t.TEAMID AND m.MAINTENANCETYPE = mt.MAINTENANCETYPE
AND c.REGISTRATIONNO = m.REGISTRATIONNO
```

```
GROUP BY mt.MAINTENANCETYPE, t.TEAMID, m.REGISTRATIONNO, c.CARBODYTYPE,
m.MAINTENANCEID , m.MAINTENANCECOST;

ALTER TABLE maintenance_temp_fact_2
ADD(BODYTYPEID NUMBER);

UPDATE maintenance_temp_fact_2
SET BODYTYPEID = 1
WHERE CARBODYTYPE = 'Bus';

UPDATE maintenance_temp_fact_2
SET BODYTYPEID = 2
WHERE CARBODYTYPE = 'Mini Bus';

UPDATE maintenance_temp_fact_2
SET BODYTYPEID = 3
WHERE CARBODYTYPE = 'People Mover';
```

	MAINTENANCETYPE	TEAMID	REGISTRATIONNO	CARBODYTYPE	MAINTENANCEID	MAINTENANCECOST	BODYTYPEID
1	M003	T005	Car21	People Mover	M220	300	3
2	M003	T002	Car14	Mini Bus	M231	300	2
3	M002	T003	Car18	Mini Bus	M238	200	2
4	M005	T005	Car14	Mini Bus	M249	500	2
5	M004	T004	Car15	Mini Bus	M257	400	2
6	M002	T005	Car16	Mini Bus	M258	200	2
7	M004	T004	Car21	People Mover	M297	400	3
8	M004	T001	Car08	Bus	M299	400	1
9	M001	T001	Car18	Mini Bus	M314	100	2
10	M003	T005	Car02	Bus	M350	300	1
11	M004	T004	Car10	Bus	M352	400	1
12	M002	T004	Car08	Bus	M353	200	1
13	M004	T002	Car23	People Mover	M359	400	3

Maintenance Fact Table

```
CREATE TABLE maintenance_fact_2 AS
SELECT MAINTENANCETYPE, BODYTYPEID, TEAMID, count(MAINTENANCEID) AS
total_maintenance_records, SUM(MAINTENANCECOST) AS
total_maintenance_cost
FROM maintenance_temp_fact_2
GROUP BY MAINTENANCETYPE, BODYTYPEID, TEAMID;
```

	MAINTENANCETYPE	BODYTYPEID	TEAMID	TOTAL_MAINTENANCE_RECORDS	TOTAL_MAINTENANCE_COST
1	M002	2	T005	23	4600
2	M004	3	T004	7	2800
3	M002	1	T004	10	2000
4	M002	2	T002	13	2600
5	M005	1	T001	19	9500
6	M005	3	T001	16	8000
7	M004	1	T005	12	4800
8	M002	2	T001	19	3800
9	M002	3	T001	12	2400
10	M004	2	T003	9	3600
11	M003	1	T003	17	5100
12	M005	2	T001	12	6000
13	M004	3	T003	11	4400

b) Star Schema Level 0 Implementation (Version-2)

DROP Table Commands

```
DROP TABLE car_dim_0;

DROP TABLE accident_temp_fact_0;

DROP TABLE accident_fact_0;

DROP TABLE maintenance_fact_0;

DROP TABLE booking_dim_0;

DROP TABLE passenger_dim_0;

DROP TABLE passenger_faculty_temp;

DROP TABLE booking_fact_0;
```

Car Dimension

```
--Car Dimension
CREATE TABLE car_dim_0 AS SELECT * FROM CAR;
```

	REGISTRATIONNO	CARMODEL	MANUFACTURINGYEAR	CARBODYTYPE	NUMSEATS
1	Car01	CapitalX	2010	Bus	40
2	Car02	CapitalX pro	2010	Bus	40
3	Car03	CapitalY	2011	Bus	40
4	Car04	CapitalY pro	2011	Bus	40
5	Car05	RunnerV1	2006	Bus	40
6	Car06	RunnerV2	2007	Bus	40
7	Car07	RunnerV3	2008	Bus	40
8	Car08	RunnerV4	2009	Bus	40

Maintenance Fact Table

```
-- Maintenance Fact Table
CREATE TABLE maintenance_fact_0 AS
SELECT mty.MAINTENANCETYPE,
        c.REGISTRATIONNO,
        mt.TEAMID,
        COUNT(MAINTENANCEID) AS total_maintenance_records,
        SUM(m.MAINTENANCECOST) AS total_maintenance_cost
FROM MAINTENANCETYPE mty,
     CAR c,
     MAINTENANCETEAM mt,
     MAINTENANCE m
WHERE mty.MAINTENANCETYPE = m.MAINTENANCETYPE
      AND c.REGISTRATIONNO = m.REGISTRATIONNO
      AND mt.TEAMID = m.TEAMID
GROUP BY  mty.MAINTENANCETYPE,
          c.REGISTRATIONNO,
          mt.TEAMID;
```

	MAINTENANCETYPE	REGISTRATIONNO	TEAMID	TOTAL_MAINTENANCE_RECORDS	TOTAL_MAINTENANCE_COST
1	M005	Car29	T004	2	1000
2	M005	Car02	T005	3	1500
3	M005	Car14	T003	4	2000
4	M005	Car10	T003	3	1500
5	M005	Car22	T001	1	500
6	M005	Car27	T004	2	1000
7	M003	Car30	T001	3	900
8	M001	Car25	T001	2	200
9	M003	Car28	T005	3	900

Accident Temp Fact table

```
-- Accident Temp Fact Table
CREATE TABLE accident_temp_fact_0 AS
SELECT
A.ACCIDENTID,
A.ACCIDENTZONE,
A.CAR_DAMAGE_SEVERITY,
E.ERRORCODE,
COUNT(A.ACCIDENTID) AS TOTAL_ACCIDENT
FROM ACCIDENTINFO A, ERROR E
WHERE A.ERRORCODE = E.ERRORCODE
GROUP BY A.ACCIDENTID, A.ACCIDENTZONE, A.CAR_DAMAGE_SEVERITY,
E.ERRORCODE;

ALTER TABLE accident_temp_fact_0
    ADD(ZONEID NUMBER);

UPDATE accident_temp_fact_0
SET ZONEID = 1
WHERE ACCIDENTZONE = 'ZoneA' ;

UPDATE accident_temp_fact_0
SET ZONEID = 2
WHERE ACCIDENTZONE = 'ZoneB' ;

UPDATE accident_temp_fact_0
SET ZONEID = 3
WHERE ACCIDENTZONE = 'ZoneC' ;

UPDATE accident_temp_fact_0
SET ZONEID = 4
WHERE ACCIDENTZONE = 'ZoneD' ;
ALTER TABLE accident_temp_fact_0
    ADD(SEVERITYID NUMBER);


UPDATE accident_temp_fact_0
SET SEVERITYID = 1
WHERE CAR_DAMAGE_SEVERITY = 'No damage';

UPDATE accident_temp_fact_0
SET SEVERITYID = 2
WHERE CAR_DAMAGE_SEVERITY = 'Very minor damage';
```

```
UPDATE accident_temp_fact_0
SET SEVERITYID = 3
WHERE CAR_DAMAGE_SEVERITY = 'Minor damage';

UPDATE accident_temp_fact_0
SET SEVERITYID = 4
WHERE CAR_DAMAGE_SEVERITY = 'Moderate damage';

UPDATE accident_temp_fact_0
SET SEVERITYID = 5
WHERE CAR_DAMAGE_SEVERITY = 'Severe damage';
```

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE	TOTAL_ACCIDENT	ZONEID	SEVERITYID
1	A647	ZoneD	No damage	Error002	1	4	1
2	A648	ZoneB	No damage	Error002	1	2	1
3	A665	ZoneA	Severe damage	Error004	1	1	5
4	A669	ZoneD	No damage	Error002	1	4	1
5	A695	ZoneB	Minor damage	Error003	1	2	3
6	A699	ZoneA	No damage	Error002	1	1	1
7	A701	ZoneD	No damage	Error002	1	4	1
8	A719	ZoneA	No damage	Error002	1	1	1
9	A722	ZoneC	Very minor damage	Error005	1	3	2
10	A737	ZoneA	Very minor damage	Error004	1	1	2
11	A742	ZoneD	Minor damage	Error005	1	4	3
12	A747	ZoneA	Severe damage	Error005	1	1	5
13	A760	ZoneD	No damage	Error002	1	4	1
14	A765	ZoneA	Moderate damage	Error003	1	1	4

Accident Fact Table

```
--Accident Fact Table
CREATE TABLE accident_fact_0 AS
SELECT ZONEID ,
SEVERITYID,
ERRORCODE,
ACCIDENTID,
COUNT(*) AS TOTAL_ACCIDENT
FROM accident_temp_fact_2
GROUP BY ZONEID,
SEVERITYID,
ERRORCODE,
ACCIDENTID;
```

	ZONEID	SEVERITYID	ERRORCODE	ACCIDENTID	TOTAL_ACCIDENT
1	1	5	Error005	A747	1
2	1	4	Error003	A765	1
3	4	4	Error003	A771	1
4	2	4	Error001	A316	1
5	2	2	Error005	A611	1
6	4	5	Error004	A1092	1
7	1	1	Error002	A1161	1
8	4	1	Error002	A851	1
9	3	5	Error001	A903	1
10	1	1	Error002	A916	1
11	3	5	Error005	A721	1
12	2	3	Error003	A423	1
13	1	4	Error003	A441	1
14	4	5	Error005	A509	1
15	1	1	Error002	A512	1

Booking Dimension

```
-- BookingDIM
CREATE TABLE booking_dim_0 AS
SELECT  BOOKINGID, BOOKINGDATE
FROM BOOKING;
```

	BOOKINGID	BOOKINGDATE
1	T716	13-JAN-19
2	T717	12-SEP-20
3	T726	26-NOV-16
4	T730	04-MAY-18
5	T736	18-DEC-21
6	T740	30-OCT-18
7	T758	13-JAN-16
8	T789	20-FEB-18
9	T797	09-SEP-19
10	T819	11-JAN-21
11	T822	08-MAR-18
12	T823	16-APR-15
13	T828	18-JUL-17
14	T835	24-MAY-15
15	T837	15-APR-19

Passenger Dimension

```
-- PassengerDIM
CREATE TABLE passenger_dim_0 AS
SELECT PASSENGERID, PASSENGERNAME, PASSENGERAGE
FROM PASSENGER;
```

	PASSENGERID	PASSENGERNAME	PASSENGERAGE
1	U020	Hope Shepard	42
2	U028	Serenity Herring	69
3	U036	Tess Sheppard	65
4	U040	Xzavier Robles	35
5	U041	Roberto Oliver	32
6	U042	Alyssa Shepherd	48
7	U043	Camden Riley	50
8	U044	Rex Kent	28
9	U045	Maximus Knox	52
10	U046	Precious Hill	43
11	U047	Gabrielle Norman	34
12	U048	Kate Hanna	34
13	U049	Elyse Martinez	39
14	U050	Renee Greer	50
15	U051	Axel Rodriguez	50

Booking Fact Table

```sql
-- Booking fact Table
CREATE TABLE passenger_faculty_temp AS
SELECT p.PASSENGERID, p.PASSENGERNAME, p.PASSENGERAGE, f.FACULTYID
FROM PASSENGER p, FACULTY f
WHERE p.FACULTYID = f.FACULTYID;
```

	PASSENGERID	PASSENGERNAME	PASSENGERAGE	FACULTYID
1	U020	Hope Shepard	42	FIT
2	U028	Serenity Herring	69	FIT
3	U036	Tess Sheppard	65	FIT
4	U040	Xzavier Robles	35	FIT
5	U041	Roberto Oliver	32	FIT
6	U042	Alyssa Shepherd	48	FIT
7	U043	Camden Riley	50	FIT
8	U044	Rex Kent	28	FIT
9	U045	Maximus Knox	52	FIT
10	U046	Precious Hill	43	FIT
11	U047	Gabrielle Norman	34	FIT
12	U048	Kate Hanna	34	FIT
13	U049	Elyse Martinez	39	FIT
14	U050	Renee Greer	50	FIT
15	U051	Axel Rodriguez	50	FIT

```sql
-- BookingFact
CREATE TABLE booking_fact_0 AS
SELECT b.BOOKINGID,  c.REGISTRATIONNO, pf.PASSENGERID, pf.FACULTYID,
COUNT(bk.BOOKINGID) AS total_booking
FROM booking_dim_0 b, passenger_faculty_temp pf, car_dim_0 c, BOOKING
bk
WHERE b.BOOKINGID = bk.BOOKINGID AND pf.PASSENGERID = bk.PASSENGERID
AND c.REGISTRATIONNO = bk.REGISTRATIONNO
GROUP BY b.BOOKINGID, pf.PASSENGERID, c.REGISTRATIONNO, pf.FACULTYID;
```

	BOOKINGID	REGISTRATIONNO	PASSENGERID	FACULTYID	TOTAL_BOOKING
1	T758	Car17	U132	FIT	1
2	T822	Car22	U120	ENG	1
3	T160	Car13	U100	SCI	1
4	T248	Car20	U153	SCI	1
5	T270	Car17	U161	SCI	1
6	T466	Car26	U121	ENG	1
7	T519	Car22	U121	ENG	1
8	T1099	Car16	U043	FIT	1
9	T1124	Car19	U140	ENG	1
10	T1191	Car03	U141	BUS	1
11	T1274	Car30	U039	ART	1
12	T1281	Car16	U031	ART	1
13	T1310	Car26	U022	BUS	1
14	T1328	Car18	U064	SCI	1
15	T1389	Car09	U115	ART	1

C.3. OLAP queries

3.1. Olap Queries

Report 1: MonCity's cumulative number of booking records of each month for Faculty of IT.

SQL command

```
SELECT f.FACULTYID,
m.MONTH,
to_char(sum(TOTAL_BOOKING), '999,999,999') AS Total_Bookings,
to_char(sum(sum(TOTAL_BOOKING)) over
    (
    PARTITION by f.FACULTYID
    ORDER BY f.FACULTYID, m.MONTH
    rows unbounded preceding), '999,999,999' ) AS Cumulative_Bookings
FROM booking_fact_2 b, month_dim_2 m, faculty_dim_2 f
WHERE b.MONTH = m.MONTH
AND b.FACULTYID = f.FACULTYID
and f.FACULTYID in ('FIT')
GROUP BY f.FACULTYID, m.MONTH;
```

Output

	FACULTYID	MONTH	TOTAL_BOOKINGS	CUMULATIVE_BOOKINGS
1	FIT	01	260	260
2	FIT	02	230	490
3	FIT	03	234	724
4	FIT	04	228	952
5	FIT	05	245	1,197
6	FIT	06	252	1,449
7	FIT	07	249	1,698
8	FIT	08	245	1,943
9	FIT	09	274	2,217
10	FIT	10	256	2,473
11	FIT	11	251	2,724
12	FIT	12	251	2,975

Report 2: MonCity's maintenance report

SQL command

```
SELECT
    DECODE(GROUPING(t.TEAMID), 1, 'All Teams', t.TEAMID) AS Team_ID,
    DECODE(GROUPING(c.CARBODYTYPE), 1, 'All Car Body Types',
c.CARBODYTYPE) AS Car_Body_Type,
    SUM(m.TOTAL_MAINTENANCE_RECORDS) AS Total_Number_Of_Maintenance,
    SUM(m.TOTAL_MAINTENANCE_COST) AS Total_Maintenance_Cost
FROM maintenance_fact_2 m, carbodytype_dim_2 c, team_dim_2 t
WHERE m.BODYTYPEID = c.BODYTYPEID
AND m.TEAMID = t.TEAMID
AND t.TEAMID IN ('T002', 'T003')
GROUP BY CUBE(t.TEAMID, c.CARBODYTYPE);
```

Output

	TEAM_ID	CAR_BODY_TYPE	TOTAL_NUMBER_OF_MAINTENANCE	TOTAL_MAINTENANCE_COST
1	All Teams	All Car Body Types	399	125300
2	All Teams	Bus	136	44900
3	All Teams	Mini Bus	113	34000
4	All Teams	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

REPORT 3: MonCity's rank analysis for the number of accidents

SQL command

```
-- REPORT 3
SELECT *
FROM
    (SELECT
        af.ERRORCODE,
        cd.REGISTRATIONNO,
        cd.CARBODYTYPE,
        SUM(af.TOTAL_ACCIDENT) AS Total_Number_Of_Accidents,
        DENSE_RANK() OVER (PARTITION BY af.ERRORCODE ORDER BY
SUM(af.TOTAL_ACCIDENT) DESC) AS RANK
    FROM
        accident_fact_2 af, accident_dim_2 ad, car_accident_bridge_2
cab, car_dim_2 cd
    WHERE
        af.ACCIDENTID = ad.ACCIDENTID
    AND ad.ACCIDENTID = cab.ACCIDENTID
    AND cab.REGISTRATIONNO = cd.REGISTRATIONNO
    GROUP BY af.ERRORCODE, cd.REGISTRATIONNO, cd.CARBODYTYPE)
WHERE RANK <= 3;
```

Output

	ERRORCODE	REGISTRATIONNO	CARBODYTYPE	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	Car04	Bus	9	3

REPORT 4: MonCity's booking report

SQL command

```
SELECT
    DECODE(GROUPING(c.CARBODYTYPE), 0, 'People Mover', c.CARBODYTYPE)
AS Car_Body_Type,
    DECODE(GROUPING(p.CATEGORYID), 1, 'All Age Groups', p.CATEGORYID)
AS Age_groups,
    DECODE(GROUPING(f.FACULTYID), 1, 'All Faculties', f.FACULTYID) AS
All_Faculties,
    SUM(b.total_booking) AS Total_Number_Of_Bookings
FROM booking_fact_2 b, carbodytype_dim_2 c, faculty_dim_2 f,
passengerage_dim_2 p
WHERE b.BODYTYPEID = c.BODYTYPEID
AND c.CARBODYTYPE IN 'People Mover'
AND p.CATEGORYID = b.CATEGORYID
AND f.FACULTYID = b.FACULTYID
AND c.CARBODYTYPE IS NOT NULL
GROUP BY c.CARBODYTYPE, CUBE(p.CATEGORYID, f.FACULTYID);
```

Output

	CAR_BODY_TYPE	AGE_GROUPS	ALL_FACULTIES	TOTAL_NUMBER_OF_BOOKINGS
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	1	All Faculties	1380
8	People Mover	1	ART	169
9	People Mover	1	BUS	121
10	People Mover	1	ENG	382
11	People Mover	1	FIT	390
12	People Mover	1	SCI	318
13	People Mover	2	All Faculties	1722
14	People Mover	2	ART	284
15	People Mover	2	BUS	193
16	People Mover	2	ENG	429
17	People Mover	2	FIT	497
18	People Mover	2	SCI	319
19	People Mover	3	All Faculties	294

Note:

Age Group 1:Young adults (18-35 years old)

Age Group 2: Middle-aged adults (36-59 years old)

Age Group 3: Old-aged adults (over 60 years old)

3.2. Reports With Rollup And Partial Rollup

Report 5

Query Question

a) What is the total number of bookings made by each passenger age group per car body type?

c) *SQL command*

```
SELECT p.CATEGORYNAME,  c.CARBODYTYPE,
       SUM(b.TOTAL_BOOKING) AS Total_Bookings
FROM booking_fact_2 b, passengerage_dim_2 p,  carbodytype_dim_2 c
WHERE b.CATEGORYID = p.CATEGORYID
AND b.BODYTYPEID = c.BODYTYPEID
GROUP BY ROLLUP( p.CATEGORYNAME, c.CARBODYTYPE);
```

d) *Output*

	CATEGORYNAME	CARBODYTYPE	TOTAL_BOOKINGS
1	Young Adults	Bus	1320
2	Young Adults	Mini Bus	1385
3	Young Adults	People Mover	1380
4	Young Adults	(null)	4085
5	Old Aged Adults	Bus	292
6	Old Aged Adults	Mini Bus	267
7	Old Aged Adults	People Mover	294
8	Old Aged Adults	(null)	853
9	Middle Aged Adults	Bus	1679
10	Middle Aged Adults	Mini Bus	1661
11	Middle Aged Adults	People Mover	1722
12	Middle Aged Adults	(null)	5062
13	(null)	(null)	10000

Report 6

Query Question

a) What is the total number of bookings made by each faculty per car body type?

b)

ROLLUP	Partial ROLLUP
Aggregated grand total value is included for all attributes.	No grand total aggregating is not included across all attributes.
Uses all attributes in the ROLLUP clause.	One or more specified attributes are taken out from the ROLLUP clause.

c) *SQL Command*

```
SELECT f.FACULTYID, c.CARBODYTYPE, SUM(b.TOTAL_BOOKING) AS
Total_Booking
FROM booking_fact_2 b, carbodytype_dim_2 c, faculty_dim_2 f
WHERE b.BODYTYPEID = c.BODYTYPEID
AND b.FACULTYID = f.FACULTYID
GROUP BY  f.FACULTYID, ROLLUP(c.CARBODYTYPE)
ORDER BY f.FACULTYID, c.CARBODYTYPE;
```

d) *Output*

	FACULTYID	CARBODYTYPE	TOTAL_BOOKING
1	ART	Bus	391
2	ART	Mini Bus	453
3	ART	People Mover	453
4	ART	(null)	1297
5	BUS	Bus	321
6	BUS	Mini Bus	337
7	BUS	People Mover	314
8	BUS	(null)	972
9	ENG	Bus	798
10	ENG	Mini Bus	769
11	ENG	People Mover	841
12	ENG	(null)	2408
13	FIT	Bus	977
14	FIT	Mini Bus	989
15	FIT	People Mover	1009
16	FIT	(null)	2975
17	SCI	Bus	804
18	SCI	Mini Bus	765
19	SCI	People Mover	779
20	SCI	(null)	2348

3.3. Report With Moving And Cumulative Aggregates

Report 7

Query Question

a) What are the total number of bookings and moving aggregate of Science faculty per month?

Importance of Data

b) The total bookings in the science faculty can help to find which month had the most bookings and the moving aggregate is useful to see the patterns in which month there was a reduction or increase in the total bookings. This can help the management to efficiently allocate more cars for the science faculty on certain months with a high number of bookings.

c) *SQL command*

```
-- REPORT 7
SELECT m.MONTH,
            to_char(SUM(b.TOTAL_BOOKING), '999,999,999') AS
Total_Bookings,
            to_char(AVG(SUM(b.TOTAL_BOOKING))
            over( order by m.MONTH rows 2 preceding), '999,999,999' )
AS Moving_Aggregate
FROM booking_fact_2 b, faculty_dim_2 f, month_dim_2 m
WHERE b.MONTH = m.MONTH
AND b.FACULTYID = f.FACULTYID
AND f.FACULTYID IN ('SCI')
GROUP BY m.MONTH ;
```

d) *Output*

	MONTH	TOTAL_BOOKINGS	MOVING_AGGREGATE
1	01	205	205
2	02	170	188
3	03	225	200
4	04	163	186
5	05	216	201
6	06	195	191
7	07	175	195
8	08	191	187
9	09	202	189
10	10	214	202
11	11	211	209
12	12	181	202

Report 8

Query Question

a) What is the cumulative maintenance cost of each car body type per maintenance team?

Importance of Data

b) The data is relevant to the management to understand how much is the cumulative maintenance cost of each car body type per maintenance team. This can help them identify which car body type and maintenance team have the highest cumulative maintenance cost and in the long run this can help them make decisions on which car body type should be cut down or operate for fewer hours in order to avoid financial crisis and significant loss of profit.

c) *SQL command*

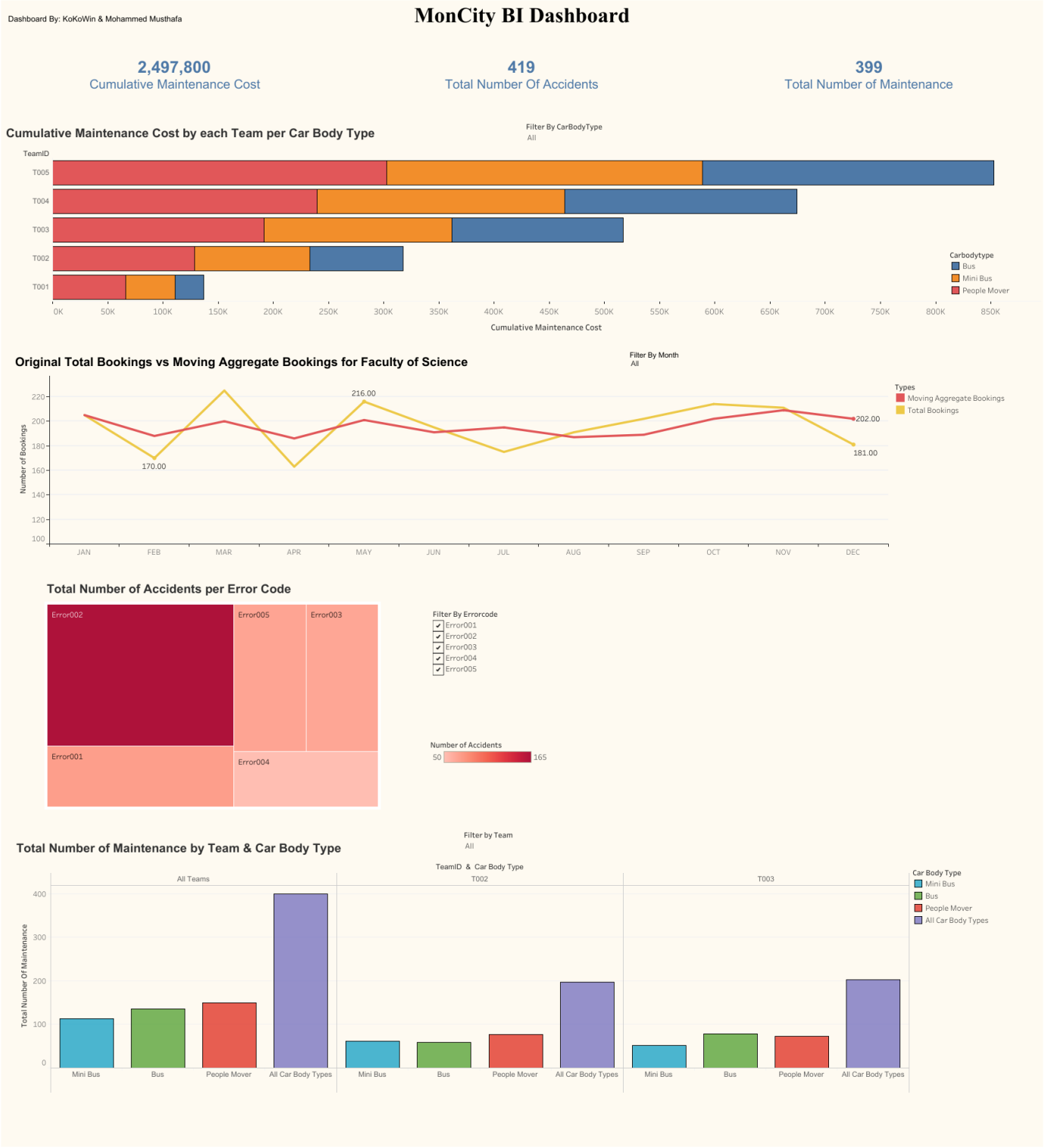
```
-- REPORT 8
SELECT
    c.CARBODYTYPE,
    m.TEAMID,
    TO_CHAR (SUM(SUM(m.TOTAL_MAINTENANCE_COST)) OVER(ORDER BY m.TEAMID,
c.CARBODYTYPE ROWS UNBOUNDED PRECEDING),'9,999,999,999') AS
CUM_MAINTENANCE_COST
FROM maintenance_fact_2 m, carbodytype_dim_2 c
WHERE m.BODYTYPEID = c.BODYTYPEID
GROUP BY c.CARBODYTYPE, m.TEAMID;
```

d) *Output*

	CARBODYTYPE	TEAMID	CUM_MAINTENANCE_COST
1	Bus	T001	25,900
2	Mini Bus	T001	44,600
3	People Mover	T001	66,200
4	Bus	T002	84,600
5	Mini Bus	T002	103,900
6	People Mover	T002	128,900
7	Bus	T003	155,400
8	Mini Bus	T003	170,100
9	People Mover	T003	191,500
10	Bus	T004	210,400
11	Mini Bus	T004	223,800
12	People Mover	T004	239,900
13	Bus	T005	263,700
14	Mini Bus	T005	286,200
15	People Mover	T005	302,700

C.4. Business Intelligence (BI) Report

Dashboard URL:
https://public.tableau.com/app/profile/kokowin/viz/MonCityBIDashboard/MonCityBI?publish=yes



C.5. Final Recommendations

First of all, from the BI dashboard it can be seen that error code **Error002** has contributed to the most number of accidents **165** while the least number of accidents occurred with error code **Error004** with only **50** accidents.

Furthermore, as for the cumulative maintenance cost of each car body type across the maintenance team it can be seen from the BI dashboard that **People Mover** has the highest maintenance cost among all car body types. Thus, it also has the highest number of maintenance of **150**. On the other hand, the maintenance cost of the **Bus** appears to be the lowest. However, it has a total maintenance number of **136** which is **23** times higher than the Mini Bus which only has a total maintenance number of **113**.

Moreover, when investigating the total booking by each faculty from the OLAP report 6 People Mover rank highest in number of bookings for **ART**, **ENG**, and **FIT** faculties. At the same time, Mini Bus has the highest number of bookings in the **ART** and **BUS** faculties. Lastly, Bus only has the highest number of bookings in the **SCI** faculty. It is clear that People Mover is a preferred choice of transport in the majority of the faculties. Furthermore, while diving deeper into total number of bookings by each month we can see that FIT faculty made highest number of bookings of **2975** follow by ENG faculty with number of bookings **2408** and SCI faculty with number of bookings of **2348**. On the other hand, ART faculty has **1297** bookings and BUS faculty has **972** bookings being the lowest among all. Moving on, as per report 7 highest number of bookings were made in March with the number of **225** bookings while April has the lowest number of **163** bookings.

To move on, when analyzing the number of accidents caused by each car body types from the OLAP report 3 it can be seen that People Mover has a total number of accidents of **165**. On the other hand, Mini Bus has **146** total accidents and the Bus has a total number of accidents **117**. Thus, due to the staggering number of accidents for People Mover, the maintenance cost and the number of maintenance for People Mover is shown to be the highest among all car body types.

Based on the data some of the suggestions to improve the self-driving car projects are as follows:

- Management needs to invest more into reducing errors for Error02 and assign more teams to work on that error.
- Management needs to ensure during March all the vehicles are not broken down or under maintenance.
- Management needs to invest more in People Mover and Mini Bus as it is the most preferred choice.
- Management needs to allocate more vehicles to FIT, ENG and SCI faculties.
- Management should perform vehicle maintenance during April as it has the lowest number of bookings.