FIT3003 : MAJOR ASSIGNMENT

MonCity – Semester 2, 2022

Rounak Agarwal & Isha Mishra

Laboratory-06_OnCampus

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
30510325	*Mishra*	**Isha**
31189555	*Agarwal*	**Rounak**

* Please include the names of all other group members.

Unit name and code	FIT3003: Business intelligence and data warehousing	
Title of assignment	Major Assignment	
Lecturer/tutor	Alice Wang	
Tutorial day and time	Friday 6pm	**Campus**
Is this an authorised group assignment? ☒ **Yes** ☐ **No**		
Has any part of this assignment been previously submitted as part of another unit/course? ☐ **Yes** ☒ **No**		
Due Date 15th October		**Date submitted 15th October**

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 The material used can be from any source (staff, students or the internet, published and unpublished works).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 .....isha....rounak..... *Date……15th october……*

* delete (iii) if not applicable

Signature _____Date:_____ Signature _____Date:_____

Signature _____Date:_____ Signature _____Date:_____

Signature _____Date:_____ 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

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
31189555	Rounak Agarwal	50%
30510325	Isha Mishra	50%

2 DECLARATION

We declare that:

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

3 SIGNATURE

Signatures

Isha Mishra

Rounak Agarwal

Date

Day Month Year

15/10 / 2022

Table of Contents

TASK C.1 : DATA WAREHOUSE FOR MONCITY DATABASE 2

A. ER DIAGRAM 2
B. DATA CLEANING 2
PART 1: RELATIONSHIPS BETWEEN TABLES 2
PART 2: CHECKING FOR NULL VALUES 3
PART 3: CHECKING FOR DUPLICATE VALUES 3
PART 4: CHECKING FOR INCORRECT VALUES 4
C. TWO VERSIONS OF STAR SCHEMA 4
VERSION 1: LEVEL 1 SCHEMA (HIGH AGGREGATION) 4
VERSION 2: LEVEL 0 SCHEMA (NO AGGREGATION) 5
D. CHOOSING DETERMINANT DIMENSION AND SCD TYPES 5
E. DIFFERENCE BETWEEN THE TWO SCHEMAS 5

TASK C.2 : IMPLEMENTATION OF THE TWO STAR SCHEMAS 5

A. SQL FOR VERSION 1 5
CREATING DIMENSIONS: 6
CREATING FACTS: 7
B. SQL FOR VERSION 2 8
CREATING DIMENSIONS: 8
CREATING FACTS: 9

TASK C.3 : REPORTS USING OLAP QUERIES 10

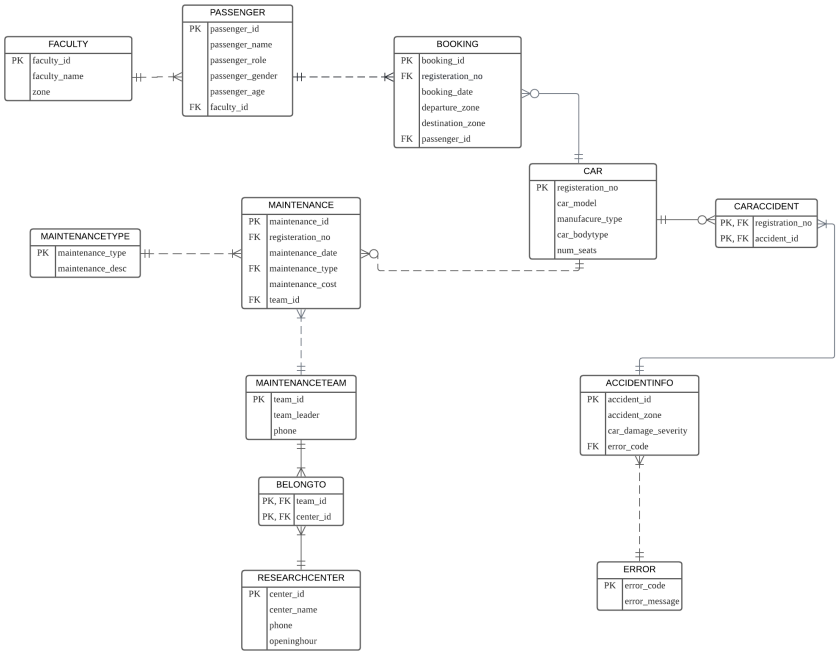
3.1. OLAP QUERIES 10
REPORT 1: MONCITY'S CUMULATIVE NUMBER OF BOOKING RECORDS OF EACH MONTH FOR FACULTY OF IT 11
REPORT 2: MONCITY'S MAINTENANCE REPORT 11
REPORT 3: MONCITY'S RANK ANALYSIS FOR THE NUMBER OF ACCIDENTS 12
REPORT 4: MONCITY'S BOOKING REPORT 12
3.2. REPORTS WITH ROLLUP AND PARTIAL ROLLUP 13
REPORT 5: TOTAL NUMBER OF BOOKINGS MADE BY DIFFERENT AGE GROUPS DURING DIFFERENT MONTHS WITHIN THE SCIENCE FACULTY 13
REPORT 6: TOTAL NUMBER OF BOOKINGS MADE BY THE YOUNG ADULTS FROM DIFFERENT FACULTY AND DURING DIFFERENT PERIODS OF THE YEAR. 14
QUESTION: DIFFERENCE BETWEEN ROLLUP AND PARTIAL ROLLUP 15
3.3. REPORT WITH MOVING AND CUMULATIVE AGGREGATES 16
REPORT 7: WHAT IS THE TWO MONTHS MOVING AVERAGE FOR THE NUMBER OF BOOKINGS MADE FOR DIFFERENT CAR BODY TYPES 16
REPORT 8: WHAT THE CUMULATED MAINTENANCE COST FOR DIFFERENT CAR BODY TYPES UNDER DIFFERENT MAINTENANCE TYPE 16
QUESTION: WHY ARE MOVING AND CUMULATIVE AGGREGATE QUERIES IMPORTANT AND VALUABLE FOR MANAGEMENT? 17

TASK C.4: BUSINESS INTELLIGENCE REPORTS 17

TASK C.5: FINAL RECOMMENDATIONS/ SUGGESTIONS 17

Task C.1 : Data Warehouse for MonCity Database

a. ER Diagram



b. Data Cleaning

The data was cleaned to remove any inconsistency.

Part 1: Relationships between tables

The PK/FK relations between the tables were checked. We found that a few tables had data for a FK that did not exist as a PK in the main table.

1. Passenger and Faculty

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

Here, “Alienware” is no faculty id in the faculty table(parent table).
SQL script used:

```
--1. passenger and faculty
SELECT
    *
FROM
    moncity.passenger
WHERE
    facultyid NOT IN (
        SELECT
            facultyid
        FROM
            moncity.faculty
    );

--creating new passenger table
DROP TABLE passengerclean CASCADE CONSTRAINTS PURGE;

CREATE TABLE passengerclean
    AS
        (
            SELECT
                *
            FROM
                moncity.passenger
            WHERE
                passengerid NOT IN (
                    SELECT
                        passenger.passengerid
                    FROM
                        moncity.passenger
                    WHERE
                        facultyid NOT IN (
                            SELECT
                                facultyid
                            FROM
                                moncity.faculty
                            )
                )
        );
```

2. Error and Accident

	ACCIDENTID	ACCIDENTZONE	CAR_DAMAGE_SEVERITY	ERRORCODE
1	A2000	ZoneB	No damage	Error010

Here, “Error010” is no error code in the error table(parent table).
SQL script used:

```
--2. Error and accident
SELECT
    *
FROM
    moncity.accidentinfo
WHERE
    errorcode NOT IN (
        SELECT
            errorcode
        FROM
            moncity.error
    );

--creating new accident info table
DROP TABLE accidentinfoclean CASCADE CONSTRAINTS PURGE;

CREATE TABLE accidentinfoclean
    AS
        (
            SELECT
                *
            FROM
                moncity.accidentinfo
            WHERE
                accidentid NOT IN (
                    SELECT
                        accidentinfo.accidentid
                    FROM
                        moncity.accidentinfo
                    WHERE
                        errorcode NOT IN (
                            SELECT
                                errorcode
                            FROM
                                moncity.accidentinfo
                        )
                )
        );
```

Part 2: Checking for Null values

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

Accident Information table had a null value. This row was decided to be deleted.
SQL script used:

```
-- checking for null values
SELECT
    *
FROM
    moncity.accidentinfo
WHERE
    accidentid IS NULL;

DELETE FROM accidentinfoclean
WHERE
    accidentid IS NULL;
```

Part 3: Checking for duplicate values

	BOOKINGID	Duplicate Count
1	T1218	2

There were duplicate values in the booking information table. Distinct values were chosen for this table.
SQL script used:

```sql
--checking for duplicate values
SELECT
    bookingid,
    COUNT(*) AS "Duplicate Count"
FROM
    moncity.booking
GROUP BY
    bookingid
HAVING
    COUNT(*) > 1;

--creating new booking table
DROP TABLE bookingclean CASCADE CONSTRAINTS PURGE;

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

Part 4: Checking for incorrect values

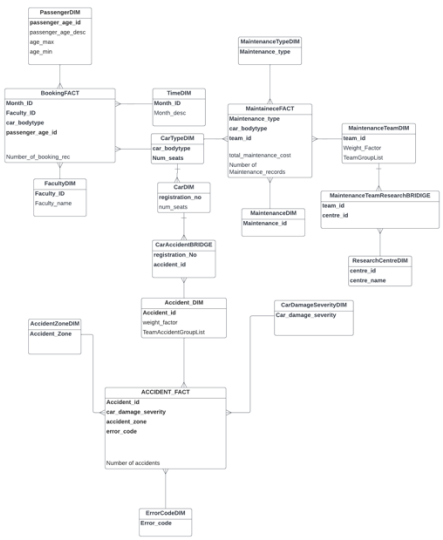
	MAINTENANCEID	REGISTRATIONNO	MAINTENANCEDATE	MAINTENANCETYPE	MAINTENANCECOST	TEAMID
1	M2000	Car13	19/JUL/15	M002	-200	T004

There was a maintenance cost lower that $0, which was incorrect. This row was deleted. SQL script used:

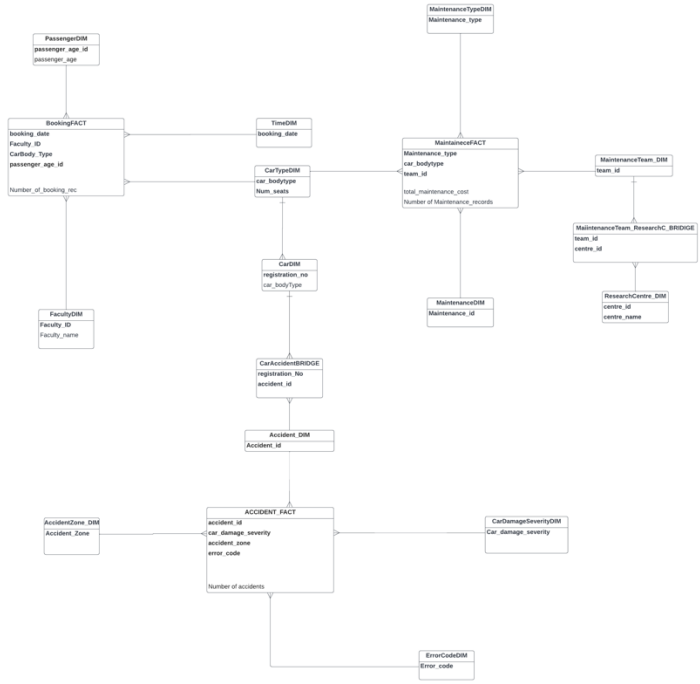
```sql
-- checking for incorrect values
SELECT
    *
FROM
    moncity.maintenance
WHERE
    maintenancecost < 0;

--creating new maintenance table
DROP TABLE maintenanceclean CASCADE CONSTRAINTS PURGE;

CREATE TABLE maintenanceclean
    AS
        (
            SELECT
                *
            FROM
                moncity.maintenance
            WHERE
                maintenancecost > 0
        );
```

c. Two Versions of Star Schema

Version 1: Level 1 schema (High Aggregation)



Version 2: Level 0 Schema (No aggregation)



d. Choosing Determinant dimension and SCD types

There are no determinant dimension. None on the facts depend on any given common dimension. Also, there was no temporal dimension. Cost of maintenance could have been temporal, but we did not find any dependency of this measure with respect to date.

e. Difference between the two schemas

Version 1 has facts broken down into aggregate functions such as passenger dimension. IN version 1, while we have the capacity of adding a rand to the age category, in level 0 we just provide with the age of the passenger. With the help of adding more aggregated functions in version 1, we get to play around with more domains and get more categories that can be analysed within our dataframe. Hence, version 2 will give more granularity and more rows within the fact table.

Task C.2 : Implementation of the two Star Schemas

a. SQL for Version 1

Creating Dimensions:

```
-- ********** Creating Maintenance Dimensions **********


-- DIM: MaintenanceTeamDIM
Drop table MaintenanceTeamDIM cascade constraints purge;

Create table MaintenanceTeamDIM as
(select t.teamid,
1/count(be.centerid) as weight_factor,
listagg(be.centerid, '_') within group (order by be.centerid) as TeamGroupList
from moncity.maintenanceteam t , moncity.belongto be
where be.teamid = t.teamid
group by t.teamid);


-- DIM: MaintenanceTypeDim
Drop table MaintenanceTypeDim cascade constraints purge;

Create table MaintenanceTypeDim as
(select maintenancetype from moncity.maintenancetype);


-- DIM: ResearchCenterDIM
Drop table ResearchCenterDIM cascade constraints purge;

Create table ResearchCenterDIM as
(select centerid, centername from moncity.researchcenter);


-- Bridge: MaintenanceResearch
Drop table MaintenanceResearchBridge cascade constraints purge;

Create table MaintenanceResearchBridge as
(select *from moncity.belongto);
```

```
-- ********** Creating Booking Dimensions **********


-- DIM: PassengerDIM
Drop table PassengerDIM cascade constraints purge;

Create table PassengerDIM
(passenger_age_id varchar2(10),
passenger_age_desc varchar2(50),
age_max number(3),
age_min number(3));

Insert into PassengerDIM values ('Group1', 'Young Adults', 18, 35);
Insert into PassengerDIM values ('Group2', 'Middle-aged Adults', 36, 59);
Insert into PassengerDIM values ('Group3', 'Old-aged Adults', 60, 120);


--DIM: FacultyDIM
Drop table FacultyDIM cascade constraints purge;

Create table FacultyDIM as
(select facultyid, facultyname
from moncity.faculty);


--DIM: TimeDIM
Drop table TimeDIM cascade constraints purge;

Create table TimeDIM as
(select distinct to_char(bookingdate, 'MM') as month_id,
to_char(bookingdate, 'Month') as month_desc
from bookingclean);
```

```
-- ********** Creating Acident Dimensions **********


--DIM: ErrorCodeDIM
Drop table ErrorCodeDIM cascade constraints purge;

create table ErrorCodeDIM as
(select errorcode from moncity.error);


--DIM: AccidentZoneDIM
Drop table AccidentZoneDIM cascade constraints purge;

create table AccidentZoneDIM as
(select distinct accidentzone from accidentinfoclean);


--DIM: CarDamageSeverityDIM
drop table CarDamageSeverityDIM cascade constraints purge;

create table CarDamageSeverityDIM as
(select distinct car_damage_severity from accidentinfoclean);


--DIM: V1_AccidentInfoDIM
Drop table V1_AccidentInfoDIM cascade constraints purge;

Create table V1_AccidentInfoDIM as
(select i.accidentid,
1/count(a.accidentid) as Weight_factor,
LISTAGG (a.registrationno, '_') within group (order by i.accidentid) as TeamAccidentGoupList
from accidentinfoclean i, moncity.caraccident a
where i.accidentid = a.accidentid
group by i.accidentid);


--BRIDGE: CarAccident
Drop table CarAccidentBridge cascade constraints purge;

create table CarAccidentBridge as
(select registrationno, accidentid from moncity.caraccident);
```

```
-- ********** Creating Shared Dimensions **********
--DIM: CarbodyDIM
DRop table CarbodyDIM cascade constraints purge;

create table CarbodyDIM as
(select distinct carbodytype, numseats from moncity.car);


--DIM: CarDIM
drop table CarDIM cascade constraints purge;

create table CarDIM as
(select distinct registrationno, carbodytype, numseats
from moncity.car);
```

Creating Facts:

```
-- ********** Creating Booking FACTS **********
-- Creating tempfact
DROP TABLE bookingtempfact CASCADE CONSTRAINTS PURGE;

CREATE TABLE bookingtempfact
    AS
        (
            SELECT
                to_char(bookingdate, 'MM') AS monthid,
                f.facultyid,
                b.bookingid,
                c.carbodytype,
                p.passengerage
            FROM
                bookingclean    b,
                passengerclean  p,
                moncity.faculty f,
                moncity.car     c
            WHERE
                    b.passengerid = p.passengerid
                AND f.facultyid = p.facultyid
                AND b.registrationno = c.registrationno
        );

ALTER TABLE bookingtempfact ADD (
    passenger_age_id VARCHAR(15)
);

UPDATE bookingtempfact
SET
    passenger_age_id = 'Group 1'
WHERE
    ( passengerage BETWEEN 18 AND 35 );

UPDATE bookingtempfact
SET
    passenger_age_id = 'Group 2'
WHERE
    ( passengerage BETWEEN 36 AND 59 );

UPDATE bookingtempfact
SET
    passenger_age_id = 'Group 1'
```

```
-- ********** Creating Maintenance FACTS **********

DROP TABLE maintenancefact CASCADE CONSTRAINTS PURGE;

CREATE TABLE maintenancefact
    AS
        (
            SELECT
                m.maintenancetype,
                carbodytype,
                m.teamid,
                COUNT(DISTINCT m.maintenanceid) AS count_records,
                SUM(m.maintenancecost)          AS maintenance_cost
            FROM
                maintenanceclean        m,
                (
                    SELECT DISTINCT
                        be.teamid
                    FROM
                        moncity.maintenanceteam,
                        moncity.belongto be,
                        moncity.researchcenter
                )                       mte,
                moncity.maintenancetype ty,
                moncity.car             c
            WHERE
                    m.teamid = mte.teamid
                AND ty.maintenancetype = m.maintenancetype
                AND c.registrationno = m.registrationno
            GROUP BY
                m.maintenancetype,
                carbodytype,
                m.teamid
        );
```

```sql
UPDATE bookingtempfact
SET
    passenger_age_id = 'Group 3'
WHERE
    ( passengerage BETWEEN 60 AND 120 );


-- Creating BookingFACT
DROP TABLE bookingfact CASCADE CONSTRAINTS PURGE;

CREATE TABLE bookingfact
    AS
        (
            SELECT
                facultyid,
                carbodytype,
                passenger_age_id,
                monthid,
                COUNT(bookingid) AS number_of_bookings
            FROM
                bookingtempfact
            GROUP BY
                facultyid,
                carbodytype,
                passenger_age_id,
                monthid
        );
```

b. SQL for Version 2

Creating Dimensions:

```sql
-- ********** Creating Maintenance Dimensions **********

-- DIM: MaintenanceDIM
DROP TABLE maintenancedim CASCADE CONSTRAINTS PURGE;

CREATE TABLE maintenancedim
    AS
        (
            SELECT
                maintenanceid
            FROM
                maintenanceclean
        );

-- DIM: MaintenanceTeamDIM
DROP TABLE maintenanceteamdim CASCADE CONSTRAINTS PURGE;

CREATE TABLE maintenanceteamdim
    AS
        (
            SELECT
                teamid
            FROM
                moncity.maintenanceteam
        );

-- DIM: MaintenanceTypeDim
DROP TABLE maintenancetypedim CASCADE CONSTRAINTS PURGE;

CREATE TABLE maintenancetypedim
    AS
        (
            SELECT
                maintenancetype
            FROM
                moncity.maintenancetype
        );
```

```sql
-- DIM: ResearchCenterDIM
DROP TABLE researchcenterdim CASCADE CONSTRAINTS PURGE;

CREATE TABLE researchcenterdim
    AS
        (
            SELECT
                centerid,
                centername
            FROM
                moncity.researchcenter
        );

-- Bridge: MaintenanceResearch
DROP TABLE maintenanceresearchbridge CASCADE CONSTRAINTS PURGE;

CREATE TABLE maintenanceresearchbridge
    AS
        (
            SELECT
                *
            FROM
                moncity.belongto
        );


-- ********** Creating Booking Dimensions **********

-- DIM: PassengerDIM
DROP TABLE passengerdim CASCADE CONSTRAINTS PURGE;

CREATE TABLE passengerdim
    AS
        (
            SELECT
                passengerid,
                passengerage
            FROM
                passengerclean
        );
```

```
--DIM: FacultyDIM
DROP TABLE facultydim CASCADE CONSTRAINTS PURGE;

CREATE TABLE facultydim
    AS
        (
            SELECT
                facultyid,
                facultyname
            FROM
                moncity.faculty
        );

--DIM: TimeDIM
DROP TABLE timedim CASCADE CONSTRAINTS PURGE;

CREATE TABLE timedim
    AS
        (
            SELECT
                bookingdate
            FROM
                bookingclean
        );


-- ********** Creating Acident Dimensions **********

--DIM: CarDIM
DROP TABLE cardim CASCADE CONSTRAINTS PURGE;

CREATE TABLE cardim
    AS
        (
            SELECT DISTINCT
                registrationno,
                carbodytype
            FROM
                moncity.car
        );
```

```
--DIM: ErrorCodeDIM
DROP TABLE errorcodedim CASCADE CONSTRAINTS PURGE;

CREATE TABLE errorcodedim
    AS
        (
            SELECT
                errorcode
            FROM
                moncity.error
        );

--DIM: AccidentZoneDIM
DROP TABLE accidentzonedim CASCADE CONSTRAINTS PURGE;

CREATE TABLE accidentzonedim
    AS
        (
            SELECT DISTINCT
                accidentzone
            FROM
                accidentinfoclean
        );

--DIM: CarDamageSeverityDIM
DROP TABLE cardamageseveritydim CASCADE CONSTRAINTS PURGE;

CREATE TABLE cardamageseveritydim
    AS
        (
            SELECT DISTINCT
                car_damage_severity
            FROM
                accidentinfoclean
        );
```

```
--DIM: CarAccidentDIM
DROP TABLE caraccidentdim CASCADE CONSTRAINTS PURGE;

CREATE TABLE caraccidentdim
    AS
        (
            SELECT
                registrationno,
                accidentid
            FROM
                moncity.caraccident
        );

--DIM: V2_AccidentInfoDIM
DROP TABLE v2_accidentinfodim CASCADE CONSTRAINTS PURGE;

CREATE TABLE v2_accidentinfodim
    AS
        (
            SELECT
                accidentid
            FROM
                accidentinfoclean
        );


-- ********** Creating Shared Dimensions **********
--DIM: CarbodyDIM
DROP TABLE carbodydim CASCADE CONSTRAINTS PURGE;

CREATE TABLE carbodydim
    AS
        (
            SELECT DISTINCT
                carbodytype,
                numseats
            FROM
                moncity.car
        );
```

Creating Facts:

```
201
202 -- ********** Creating Maintenance FACTS **********
203
204 DROP TABLE maintenancefact CASCADE CONSTRAINTS PURGE;
205
206 CREATE TABLE maintenancefact
207     AS
208         (
209             SELECT
210                 m.maintenanceid,
211                 m.maintenancetype,
212                 c.carbodytype,
213                 m.teamid,
214                 COUNT(DISTINCT maintenanceid) AS record_numbers,
215                 SUM(maintenancecost)          AS total_cost
216             FROM
217                 maintenanceclean        m,
218                 (
219                     SELECT DISTINCT
220                         be.teamid
221                     FROM
222                         moncity.maintenanceteam,
223                         moncity.belongto be,
224                         moncity.researchcenter
225                 )                           mte,
226                 moncity.maintenancetype ty,
227                 moncity.car             c
228             WHERE
229                     m.teamid = mte.teamid
230                 AND ty.maintenancetype = m.maintenancetype
231                 AND c.registrationno = m.registrationno
232             GROUP BY
233                 m.maintenancetype,
234                 c.carbodytype,
235                 m.teamid,
236                 m.maintenanceid
237         );
238
239
```

```
239
240 -- ********** Creating Accident FACTS **********
241 DROP TABLE accidentfact CASCADE CONSTRAINTS PURGE;
242
243 CREATE TABLE accidentfact
244     AS
245         (
246             SELECT
247                 accidentid,
248                 accidentzone,
249                 car_damage_severity,
250                 errorcode,
251                 COUNT(accidentid) AS accident_numbers
252             FROM
253                 accidentinfoclean
254             GROUP BY
255                 accidentid,
256                 accidentzone,
257                 car_damage_severity,
258                 errorcode
259         );
260
```

```
260
261 -- ********** Creating Booking FACTS **********
262 DROP TABLE bookingfact CASCADE CONSTRAINTS PURGE;
263
264 CREATE TABLE bookingfact
265     AS
266         (
267             SELECT
268                 f.facultyid,
269                 c.carbodytype,
270                 p.passengerage,
271                 b.bookingdate,
272                 b.bookingid,
273                 COUNT(b.bookingid) AS number_of_bookings
274             FROM
275                 bookingclean    b,
276                 passengerclean  p,
277                 moncity.car     c,
278                 moncity.faculty f
279             WHERE
280                     b.registrationno = c.registrationno
281                 AND f.facultyid = p.facultyid
282                 AND p.passengerid = b.passengerid
283             GROUP BY
284                 f.facultyid,
285                 b.bookingid,
286                 c.carbodytype,
287                 p.passengerage,
288                 b.bookingdate
289         );
```

Task C.3 : Reports Using OLAP queries

3.1. OLAP Queries

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

a. SQL script used

```sql
SELECT
    b.facultyid,
    t.month_desc                AS "Month",
    SUM(b.number_of_bookings) AS "Total Bookings",
    to_char(SUM(SUM(number_of_bookings))
            OVER(
        ORDER BY
            b.facultyid,
            TO_DATE(t.month_desc, 'Month')
        ROWS UNBOUNDED PRECEDING
            ),
            '9,999,999')       AS "Cumulative number of booking records"
FROM
    bookingfact b,
    timedim     t
WHERE
        b.monthid = t.month_id
    AND b.facultyid = 'FIT'
GROUP BY
    b.facultyid,
    t.month_desc
ORDER BY
    MIN(t.month_id);
```

b. Result

	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

Report 2: MonCity's maintenance report

a. SQL script used

```sql
SELECT
    decode(GROUPING(teamid),
           1,
           'All Teams',
           teamid)       AS "Team ID",
    decode(GROUPING(carbodytype),
           1,
           'All Car Body Types',
           carbodytype)  AS "Car body type",
    SUM(count_records)   AS "Total number of maintenance",
    to_char(SUM(maintenance_cost),
            '9,999,999') AS "Total maintenance cost"
FROM
    maintenancefact
WHERE
    teamid IN ( 'T002', 'T003' )
GROUP BY
    CUBE(teamid,
         carbodytype);
```

b. Result

	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

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

a. SQL script used

```sql
SELECT
    *
FROM
    (
        SELECT
            f.errorcode          AS "Error Code",
            c.registrationno     AS "Registration No.",
            c.carbodytype        AS "Car Body Type",
            COUNT(f.accidentid) AS "Total Number of accidents",
            DENSE_RANK()
            OVER(PARTITION BY f.errorcode
                 ORDER BY
                     COUNT(f.accidentid) DESC
            )                    AS rank
        FROM
            accidentfact      f,
            accidentinfoclean       a,
            caraccidentbridge b,
            cardim            c
        WHERE
                f.accidentid = a.accidentid
            AND b.accidentid = f.accidentid
            AND c.registrationno = b.registrationno
        GROUP BY
            f.errorcode,
            c.registrationno,
            c.carbodytype
        ORDER BY
            f.errorcode
    )
WHERE
    rank <= 3;
```

b. Result

	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

Report 4: MonCity's booking report

a. SQL script used

```sql
SELECT
    carbodytype               AS "Car body type",
    decode(GROUPING(passenger_age_id),
           1,
           'All Age groups',
           passenger_age_id) AS "Age Group",
    decode(GROUPING(facultyid),
           1,
           'All faculties',
           facultyid)         AS "Faculty ID",
    to_char(SUM(number_of_bookings),
            '99,999,999')     AS "Total number of bookings"
FROM
    bookingfact
WHERE
    carbodytype = 'People Mover'
GROUP BY
    carbodytype,
    CUBE(passenger_age_id,
         facultyid);
```

b. Result

	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	Group 1	All faculties	1,674
8	People Mover	Group 1	ART	169
9	People Mover	Group 1	BUS	121
10	People Mover	Group 1	ENG	412
11	People Mover	Group 1	FIT	512
12	People Mover	Group 1	SCI	460
13	People Mover	Group 2	All faculties	1,722
14	People Mover	Group 2	ART	284
15	People Mover	Group 2	BUS	193

3.2. Reports with rollup and partial rollup

Report 5: Total number of bookings made by different age groups during different months within the Science Faculty

c. SQL script used

```
SELECT
    decode(GROUPING(facultyname),
           1,
           'All Faculty',
           facultyname)          AS "Faculty Name",
    decode(GROUPING(p.passenger_age_desc),
           1,
           'All Age Groups',
           p.passenger_age_desc) AS "Age Group",
    decode(GROUPING(month_desc),
           1,
           'All Months',
           month_desc)           AS "Months",
    lpad(to_char(SUM(number_of_bookings),
                 '99,999'),
         20,
         ' ')                    AS "Total number of bookings"
FROM
    bookingfact  b,
    timedim      t,
    passengerdim p,
    facultydim   f
WHERE
        b.monthid = t.month_id
    AND b.facultyid = f.facultyid
    AND p.passenger_age_id = b.passenger_age_id
GROUP BY
    ROLLUP(facultyname,
           p.passenger_age_desc,
           t.month_desc);
```

d. Result

	Faculty Name	Age Group	Months	Total number of bookings
1	Science	Young adult	APRIL	57
2	Science	Young adult	AUGUST	65
3	Science	Young adult	DECEMBER	63
4	Science	Young adult	FEBRUARY	68
5	Science	Young adult	JANUARY	91
6	Science	Young adult	JULY	72
7	Science	Young adult	JUNE	81
8	Science	Young adult	MARCH	96
9	Science	Young adult	MAY	89
10	Science	Young adult	NOVEMBER	91
11	Science	Young adult	OCTOBER	103
12	Science	Young adult	SEPTEMBER	87
13	Science	Young adult	All Months	963
14	Science	Middle adult	APRIL	72
15	Science	Middle adult	AUGUST	93
16	Science	Middle adult	DECEMBER	88
17	Science	Middle adult	FEBRUARY	67
18	Science	Middle adult	JANUARY	87
19	Science	Middle adult	JULY	68
20	Science	Middle adult	JUNE	74
21	Science	Middle adult	MARCH	100
22	Science	Middle adult	MAY	94
23	Science	Middle adult	NOVEMBER	85
24	Science	Middle adult	OCTOBER	80
25	Science	Middle adult	SEPTEMBER	77
26	Science	Middle adult	All Months	985
27	Science	Old-aged adult	APRIL	34
28	Science	Old-aged adult	AUGUST	33
29	Science	Old-aged adult	DECEMBER	30
30	Science	Old-aged adult	FEBRUARY	35
31	Science	Old-aged adult	JANUARY	27

Report 6: Total number of bookings made by the Young adults from different faculty and during different periods of the year.

c. SQL Script used

```
35
36  SELECT
37      decode(GROUPING(p.passenger_age_desc),
38             1,
39             'All Age Groups',
40             p.passenger_age_desc) AS "Age Group",
41      decode(GROUPING(facultyname),
42             1,
43             'All Faculty',
44             facultyname)          AS "Faculty Name",
45      decode(GROUPING(month_desc),
46             1,
47             'All Months',
48             month_desc)           AS "Months",
49      lpad(to_char(SUM(number_of_bookings),
50                   '99,999'),
51           20,
52           ' ')                    AS "Total number of bookings"
53  FROM
54      bookingfact  b,
55      timedim      t,
56      passengerdim p,
57      facultydim   f
58  WHERE
59          b.monthid = t.month_id
60      AND b.facultyid = f.facultyid
61      AND p.passenger_age_id = b.passenger_age_id
62  GROUP BY
63      p.passenger_age_desc,
64      ROLLUP(t.month_desc,
65             facultyname);
```

d. Result

	Age Group	Faculty Name	Months	Total number of bookings
1	Young adult	Science	APRIL	57
2	Young adult	Engineering	APRIL	93
3	Young adult	Business and Economics	APRIL	20
4	Young adult	Information Technology	APRIL	95
5	Young adult	Art, Design and Architecture	APRIL	47
6	Young adult	All Faculty	APRIL	312
7	Young adult	Science	AUGUST	65
8	Young adult	Engineering	AUGUST	85
9	Young adult	Business and Economics	AUGUST	28
10	Young adult	Information Technology	AUGUST	85
11	Young adult	Art, Design and Architecture	AUGUST	45
12	Young adult	All Faculty	AUGUST	308
13	Young adult	Science	DECEMBER	63
14	Young adult	Engineering	DECEMBER	101
15	Young adult	Business and Economics	DECEMBER	28
16	Young adult	Information Technology	DECEMBER	101
17	Young adult	Art, Design and Architecture	DECEMBER	46
18	Young adult	All Faculty	DECEMBER	339
19	Young adult	Science	FEBRUARY	68
20	Young adult	Engineering	FEBRUARY	78
21	Young adult	Business and Economics	FEBRUARY	27
22	Young adult	Information Technology	FEBRUARY	81
23	Young adult	Art, Design and Architecture	FEBRUARY	40
24	Young adult	All Faculty	FEBRUARY	294
25	Young adult	Science	JANUARY	91
26	Young adult	Engineering	JANUARY	114
27	Young adult	Business and Economics	JANUARY	29
28	Young adult	Information Technology	JANUARY	114
29	Young adult	Art, Design and Architecture	JANUARY	47

Question: Difference between Rollup and Partial Rollup

Answer: Rollup queries are helpful to calculate a number of subtotals across a given dimension. In our case we are rolling up the number of bookings. This helps in finding the grand total over a given period. This can not be achieved in partial roll up. Partial roll up is used to find partial sums.

3.3. Report with moving and cumulative aggregates

Report 7: What is the two months moving average for the number of bookings made for different car body types

c. SQL script used

```
SELECT
    monthid,
    carbodytype,
    SUM(number_of_bookings) AS "Number of bookings",
    to_char(AVG(SUM(number_of_bookings))
                OVER(
            ORDER BY
                monthid
            ROWS 2 PRECEDING
                ),
                '9,999,999.99') AS "Two Months Moving Average"
FROM
    bookingfact
GROUP BY
    monthid,
    carbodytype;
```

d. Result

	MONTHID	CARBODYTYPE	Number of bookings	Two Months Moving Average
1	01	Bus	291	291.00
2	01	Mini Bus	293	292.00
3	01	People Mover	300	294.67
4	02	Bus	279	290.67
5	02	Mini Bus	211	263.33
6	02	People Mover	261	250.33
7	03	Bus	260	244.00
8	03	Mini Bus	289	270.00
9	03	People Mover	302	283.67
10	04	Bus	250	280.33
11	04	Mini Bus	264	272.00
12	04	People Mover	273	262.33
13	05	Bus	281	272.67
14	05	Mini Bus	264	272.67
15	05	People Mover	288	277.67

Report 8: What the cumulated maintenance cost for different car body types under different maintenance type

c. SQL script used

```
SELECT
    maintenancetype,
    carbodytype,
    SUM(count_records)        AS "Total number of maintenance Record",
    to_char(SUM(maintenance_cost),
                '9,999,999,999') AS "Total maintenance cost",
    to_char(SUM(SUM(maintenance_cost))
                OVER(
            ORDER BY
                maintenancetype,
                carbodytype
            ROWS UNBOUNDED PRECEDING
                ),
                '9,999,999,999') AS "Cummulative total maintenance cost"
FROM
    maintenancefact
GROUP BY
    maintenancetype,
    carbodytype;
```

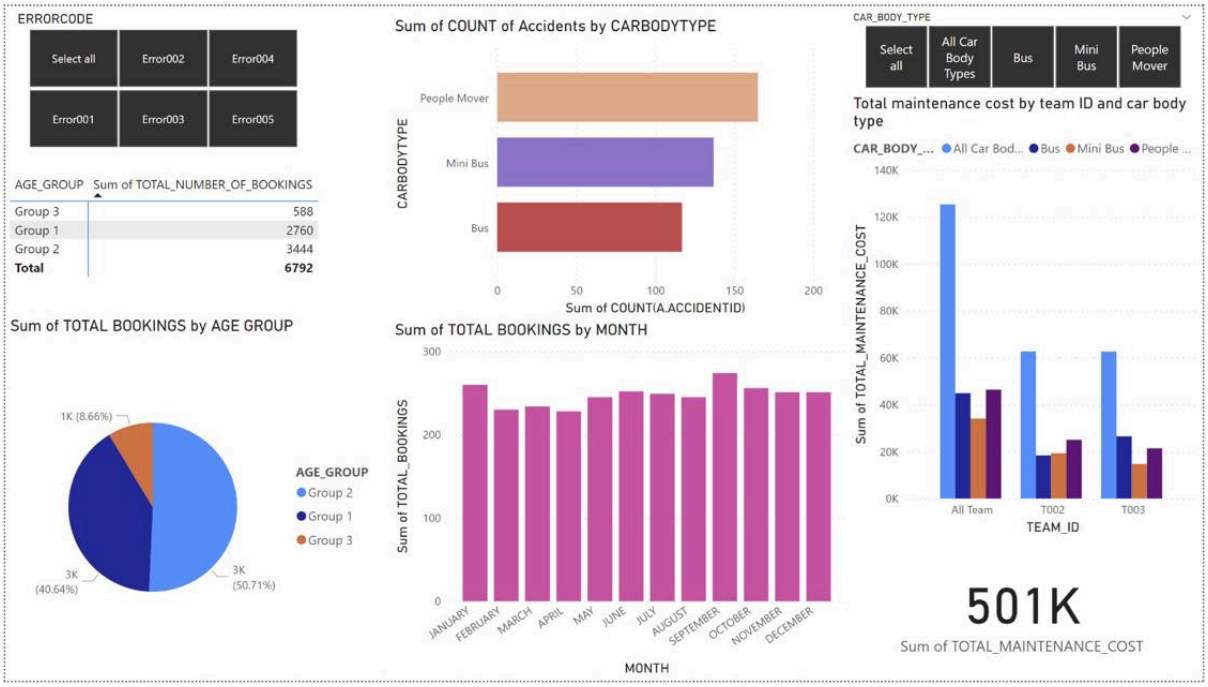
d. Result

	MAINTENANCETYPE	CARBODYTYPE	Total number of maintenance Record	Total maintenance cost	Cummulative total maintenance cost
1	M001	Bus	56	5,600	5,600
2	M001	Mini Bus	73	7,300	12,900
3	M001	People Mover	75	7,500	20,400
4	M002	Bus	66	13,200	33,600
5	M002	Mini Bus	71	14,200	47,800
6	M002	People Mover	60	12,000	59,800
7	M003	Bus	74	22,200	82,000
8	M003	Mini Bus	59	17,700	99,700
9	M003	People Mover	64	19,200	118,900
10	M004	Bus	70	28,000	146,900
11	M004	Mini Bus	46	18,400	165,300
12	M004	People Mover	56	22,400	187,700
13	M005	Bus	89	44,500	232,200
14	M005	Mini Bus	62	31,000	263,200
15	M005	People Mover	79	39,500	302,700

Question: Why are moving and cumulative aggregate queries important and valuable for management?

Answer: With the help of theses aggregate queries we are able to see the history of the maintenance cost. It is an important indicator to see the movement over time. In report 8, we can see how different car body types influence the number of maintenance they need to undergo. This can help reviewing the manufacturing failures within the body type. Whereas in report 7, we can see the total number of booking people make on the basis of the car body type. This can help in showing how the friend circle/ family size is evolving for people.

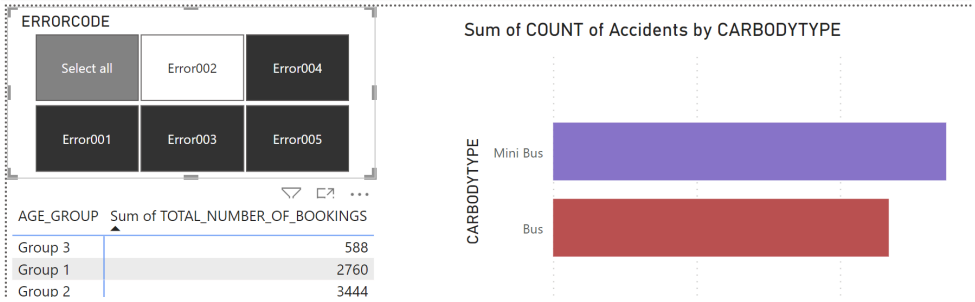
Task C.4: Business Intelligence Reports



Task C.5: Final Recommendations/ Suggestions

It can be seen from the Pie chart (Total number of bookings by age group) that there are the highest numbers of bookings made for the age groups 1 and 2 (close to 91% of all the bookings made). As a result, I would recommend that rather than having separate buses for people above 60 years of age, we should have reserved seats for them, which would save both time and money, instead of having separate buses for these people. According to the

chart (which shows the sum of the number of accidents by car type) and using the filter, it is evident that only Error 2 can cause an accident with a People mover car body type, based on the data shown in the chart. In addition, it is evident from the chart (total maintenance cost based on car body type) that the People Mower requires the highest level of maintenance compared to other vehicle types. In light of the fact that Error 2 is related to Low battery, it would be beneficial to improve them in order to fix it.



As we can see from the graph sum indicating the total number of bookings by month, it is noticeable that the most bookings are made during the month of January and September. We could suggest having some more people movers or increasing the frequency of other buses during these months.