FIT5137 Group Assignment

OPEN FLIGHTS & TRAVEL DATA WAREHOUSE

Yaling Xiong, Yizheng Fang

Monash University

Table of Contents

Details

1. Oracle accounts

Account name: S29408152(yaling xiong), S29407699(yizheng fang)
Account password: student

2. Contribution declaration form

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
29401852	Yaling Xiong	50%
29407699	Yizheng Fang	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 LIST OF PARTS THAT EACH STUDENT DID

List of parts that each student did

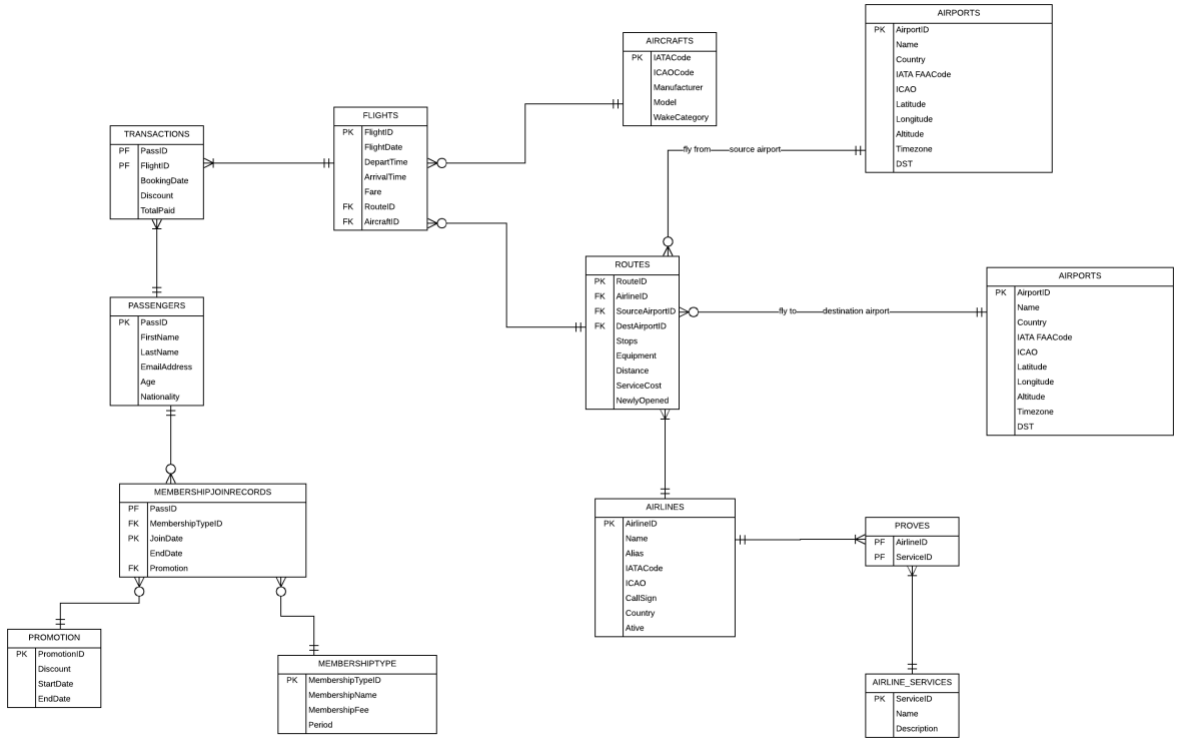
- Yizheng Fang (29407699): Task C.1, Task C.2.a, Task C.5
- Yaling Xiong (29401852): Task C.2.b, Task C.3, Task C.4, Task C.5

3 SIGNATURE

	Yaling Xiong	Yizheng Fang
Signatures		
Date	Day Month Year 14 / 10 / 2018	

Task C.1(preparation stage)

a. The E/R diagram of the operational database.



b. Data Cleaning

We do data cleaning based on each entity. The questions we check are:
1. check if there are duplicate rows
2. check if primary key is unique and not null.
3. check referential integrity violent.
4. check if there is missing value
5. check invalid values (i.e ID number should not be negative)

1. Transactions Entity

Explore the Data by general view of data using
SELECT * FROM TRANSACTIONS FETCH FIRST 10 ROWS ONLY;

```
    PASSID FLIGHTID   BOOKINGDA   DISCOUNT  TOTALPAID
---------- ---------- --------- ---------- ----------
      1063 TL4217     02/SEP/09          0     530.42
      3422 JL7937     26/APR/07          0    1759.93
      4304 DE2739     03/FEB/09        .09    1257.55
      8911 JQ5983     14/JUN/07        .02     333.34
      1655 QF9865     19/AUG/08          0     172.43
      5905 Q65229     30/AUG/06        .03     656.65
      6623 GA2888     19/APR/07          0     831.61
      1290 SH4537     27/APR/09          0     522.86
     10962 FQ5919     02/JUN/06        .03     282.41
      2640 BR1164     08/FEB/09          0     836.78

10 rows selected.
```

Count the total number of transactions records existing in the entity which is 25005 records.

```
SELECT COUNT (*) TOTAL_TRANSACTION FROM TRANSACTIONS;
```

```
TOTAL_TRANSACTION
-----------------
            25005
```

Check any null values in the transaction entity, and there are no null values in transactions attributes.

```
SELECT COUNT (*) AS PASSID_NULL FROM TRANSACTIONS
WHERE PASSID IS NULL;

SELECT COUNT (*) AS FLIGHTID_NULL FROM TRANSACTIONS
WHERE FLIGHTID IS NULL;

SELECT COUNT (*) AS BOOKINGDATE_NULL FROM TRANSACTIONS
WHERE BOOKINGDATE IS NULL;

SELECT COUNT (*) AS DISCOUNT_NULL FROM TRANSACTIONS
WHERE DISCOUNT IS NULL;

SELECT COUNT (*) AS TOTALPAID_NULL FROM TRANSACTIONS
WHERE TOTALPAID IS NULL;
```

```
PASSID_NULL
-----------
          0

FLIGHTID_NULL
-------------
           0

BOOKINGDATE_NULL
----------------
              0

DISCOUNT_NULL
-------------
           0

TOTALPAID_NULL
--------------
            0
```

Then we need to figure out whether there is any negative values or extreme values in totalpaid attribute, there are no extreme or negative values, the max totalpaid is 345.38 and the min totalpaid is 74.04.

```
SELECT COUNT (*) FROM TRANSACTIONS
WHERE TOTALPAID<0;
```

```
  COUNT(*)
----------
         0
```

```
SELECT MAX(TOTALPAID), MIN(TOTALPAID) FROM TRANSACTIONS;
```

```
MAX(TOTALPAID) MIN(TOTALPAID)
-------------- --------------
       3459.38          74.04
```

Furthermore, check the discount attribute values, and there are no negative or extreme values, the max discount is 10% and min discount is 0.

```
SELECT MAX(DISCOUNT), MIN(DISCOUNT)
FROM TRANSACTIONS;
```

```
MAX(DISCOUNT) MIN(DISCOUNT)
------------- -------------
           .1             0
```

Meanwhile, we need to check the booking date is within the required date range : during the period 01-01-2006 and 31-12-2009, and we find that there are 5 records are beyond the required date range, which might need to be fixed.

```
SELECT (TO_CHAR(BOOKINGDATE,'DD-MM-YYYY')) FROM TRANSACTIONS
WHERE TO_CHAR(BOOKINGDATE,'YY') > '09'
ORDER BY BOOKINGDATE DESC;
```

```
BOOKINGDAT
----------
05-04-2015
05-04-2015
05-04-2015
05-04-2015
05-04-2015
```

Before we drop the invalid records , we need to figure out whether there is any problem in reference table, flights entity. We find there are some mismatch records and it may be just caused by the dates out of range.

```
SELECT T. FLIGHTID AS TF, F. FLIGHTID AS FF, T.BOOKINGDATE
FROM TRANSACTIONS T FULL OUTER JOIN FLIGHTS F ON T. FLIGHTID = F. FLIGHTID
WHERE F. FLIGHTID IS NULL;
```

```
TF         FF         BOOKINGDA
---------- ---------- ---------
GHOST5                05/APR/15
GHOST1                05/APR/15
GHOST4                05/APR/15
GHOST2                05/APR/15
GHOST3                05/APR/15
```

Then check duplicates records, using combination of primary key PassId and FlightId, there is no duplicated record in transactions entity.

```
SELECT PASSID, FLIGHTID, COUNT (*) FROM TRANSACTIONS
GROUP BY PASSID, FLIGHTID
HAVING COUNT (*) > 1;
```

```
no rows selected
```

Now, it's time to remove the incorrect or invalid records from the database.

```
DELETE FROM TRANSACTIONS
WHERE TO_CHAR(BOOKINGDATE,'YY') > '09';
```

Then after dropping the records, the total records become 25000.

```
SELECT COUNT (*) TOTAL_TRANSACTION FROM TRANSACTIONS;
```

And there is no record that beyond the given date range.

```
SELECT TO_CHAR(BOOKINGDATE,'DD-MM-YYYY') AS BEYOND_DATE FROM TRANSACTIONS
WHERE TO_CHAR(BOOKINGDATE,'YY') > '09';
```

```
5 rows deleted.

TOTAL_TRANSACTION
-----------------
            25000

no rows selected
```

2. Flights Entity

Having a look at the records number of flights, and there are 50002 records in flights entity.

```
SELECT COUNT(*) FROM FLIGHTS;
```

```
  COUNT(*)
----------
     50002
```

Checking the null values in each attribute in flights entity, and there is no null value.

```
SELECT COUNT(*) AS FLIGHT_NULL FROM FLIGHTS
WHERE FLIGHTID IS NULL;

SELECT COUNT(*) AS FLIGHTDATE_NULL FROM FLIGHTS
WHERE FLIGHTDATE IS NULL;

SELECT COUNT(*) AS DEPARTTIME_NULL FROM FLIGHTS
WHERE DEPARTTIME IS NULL;

SELECT COUNT(*) AS ARRIVALTIME_NULL FROM FLIGHTS
WHERE ARRIVALTIME IS NULL;

SELECT COUNT(*) AS FARE_NULL FROM FLIGHTS
WHERE FARE IS NULL;

SELECT COUNT(*) AS ROUTEID_NULL FROM FLIGHTS
WHERE ROUTEID IS NULL;

SELECT COUNT(*) AS AIRCRAFTID_NULL FROM FLIGHTS
WHERE AIRCRAFTID IS NULL;
```

```
FLIGHT_NULL
-----------
          0

FLIGHTDATE_NULL
---------------
              0

DEPARTTIME_NULL
---------------
              0

ARRIVALTIME_NULL
----------------
               0

 FARE_NULL
----------
         0

ROUTEID_NULL
------------
           0

AIRCRAFTID_NULL
---------------
              0
```

Besides, make sure that the primary key, flightId is unique, in other words, no duplicate records. There are no duplicate records for flights entity.

```
SELECT FLIGHTID, COUNT (*) FROM FLIGHTS
GROUP BY FLIGHTID
HAVING COUNT (*) > 1;
```

```
no rows selected
```

Check Aircraftid valid or not, Aircraftid should be 3 letter length. Meanwhile, the Aircraftid is also named as IATACode in Aircraft entity. There are two records that have aircraftid with length 4 and we noticed that the fare has invalid data with negative value.

```
SELECT * FROM FLIGHTS
WHERE LENGTH(TRIM(AIRCRAFTID))! =3;
```

FLIGHTID	FLIGHTDAT	DEPARTTIM	ARRIVALTI	FARE	ROUTEID	AIRC
ABCDEF	05/APR/15	05/APR/15	05/APR/15	99.99	56	5137
DREAMS	05/APR/15	05/APR/15	05/APR/15	-99.99	56	5137

Check fare max and min. There is a negative number which should be removed. But check the max value in case there are the extreme values.

```
SELECT MAX(FARE), MIN(FARE) FROM FLIGHTS;
```

MAX(FARE)	MIN(FARE)
1818.01	-99.99

Check negative value in flights entity

```
SELECT * FROM FLIGHTS WHERE FARE<0;
```

FLIGHTID	FLIGHTDAT	DEPARTTIM	ARRIVALTI	FARE	ROUTEID	AIRC
DREAMS	05/APR/15	05/APR/15	05/APR/15	-99.99	56	5137

Remove negative numbers in fare attribute in flights entity. Only one record is removed.

```
DELETE FROM FLIGHTS WHERE FARE<0;
```

After cleaning negative number in fare attribute in flights entity

```
SELECT COUNT(*) FROM FLIGHTS WHERE FARE<0;
```

```
1 row deleted.

  COUNT(*)
----------
         0
```

Check the referential integrity using flights entity full outer join on aircraft entity, and we find there is one record that is empty which is also have problem with length of aircraftId. Also, we need to check referential integrity between flights and transaction. All invalid flights are not in transactions entity.

```
SELECT * FROM TRANSACTIONS WHERE FLIGHTID IN
(SELECT FLIGHTID
FROM FLIGHTS F FULL OUTER JOIN AIRCRAFTS A ON F.AIRCRAFTID = A.IATACODE
WHERE A.IATACODE IS NULL);
```

Check referential integrity violation, all invalid flights are not in routes entity.

```
SELECT * FROM FLIGHTS F FULL OUTER JOIN ROUTES R ON F.ROUTEID = R.ROUTEID
WHERE R.ROUTEID IS NULL;
```

```
no rows selected
no rows selected
```

Check all records are in valid range of dates before remove any records, and the record which is not within valid date range has the invalid aircraftId

```
SELECT * FROM FLIGHTS
WHERE TO_CHAR(FLIGHTDATE,'YY') > '12';
```

FLIGHTID	FLIGHTDAT	DEPARTTIM	ARRIVALTI	FARE	ROUTEID	AIRC
ABCDEF	05/APR/15	05/APR/15	05/APR/15	99.99	56	5137

Remove invalid aircraftId

```
DELETE FROM FLIGHTS
WHERE LENGTH(TRIM(AIRCRAFTID))! =3;
```

Check again using SQL Command.

```
SELECT * FROM FLIGHTS
```

```
WHERE LENGTH(TRIM(AIRCRAFTID))! =3;
1 row deleted.

  COUNT(*)
----------
         0
```

3. Passengers Entity

Check age max and min number, the oldest passenger is 87 years old and the youngest is -1 which is not acceptable.

```
SELECT MAX(AGE), MIN(AGE) FROM PASSENGERS;
  MAX(AGE)   MIN(AGE)
---------- ----------
        87         -1
```

Check the negative values for age and there are four passengers have negative age which are not valid for our data.

```
SELECT * FROM PASSENGERS WHERE AGE<0;
```

PASSID	FIRSTNAME	LASTNAME	EMAILADDRESS	AGE	NATIONALITY
1000	Wayne	Rooney	wayne@Gmail.com	-1	English
999	Juan	Mata	juan@Gmail.com	-1	Spainish
998	Adan	Januzaj	adan@Gmail.com	-1	Belgian
997	Luke	Shaw	luke@Gmail.com	-1	English
996	Phil	Jones	phil@Gmail.com	-1	English

Before deleting invalid data, check if they are in transactions entities

```
SELECT * FROM TRANSACTIONS WHERE PASSID IN
(SELECT PASSID FROM PASSENGERS WHERE AGE<0);
```

Check if they are in membershipjoinrecords entity

```
SELECT * FROM MEMBERSHIPJOINRECORDS
WHERE PASSID IN (SELECT PASSID FROM PASSENGERS WHERE AGE<0);
```

check duplicates PassId(records), no duplicates

```
SELECT PASSID, COUNT (*) FROM PASSENGERS
GROUP BY PASSID
HAVING COUNT(PASSID)>1;
no rows selected
no rows selected
no rows selected
```

check duplicate names

```
SELECT FIRSTNAME,LASTNAME,COUNT(*)  FROM PASSENGERS
GROUP BY FIRSTNAME,LASTNAME
HAVING COUNT(*)>1;
```

FIRSTNAME	LASTNAME	COUNT(*)
Lorenzo	Nickolas	2
Florine	Julieann	2

double check, they are valid records

```
SELECT *
FROM PASSENGERS
WHERE FIRSTNAME='LORENZO' AND LASTNAME='NICKOLAS';
SELECT *
FROM PASSENGERS
WHERE FIRSTNAME='FLORINE' AND LASTNAME='JULIEANN';
```

PASSID	FIRSTNAME	LASTNAME	EMAILADDRESS
8108	Lorenzo	Nickolas	Lorenzo.Nickolas@hotmail.com
8779	Lorenzo	Nickolas	Lorenzo.Nickolas@yahoo.com

PASSID	FIRSTNAME	LASTNAME	EMAILADDRESS
1939	Florine	Julieann	Florine.Julieann@yahoo.com
5888	Florine	Julieann	Florine.Julieann@mail.com

all email addresses are unique

```
SELECT EMAILADDRESS, COUNT (*)
FROM PASSENGERS
GROUP BY EMAILADDRESS
HAVING COUNT (*)>1;
```

```
no rows selected
```

check null values, there is no missing records in the attributes of passengers entity.

```
SELECT COUNT (*) FROM PASSENGERS WHERE PASSID IS NULL;

SELECT COUNT(*) FROM PASSENGERS WHERE AGE IS NULL;

SELECT COUNT(*) FROM PASSENGERS WHERE NATIONALITY IS NULL;

SELECT COUNT(*) FROM PASSENGERS WHERE EMAILADDRESS IS NULL;

SELECT COUNT(*) FROM PASSENGERS WHERE FIRSTNAME IS NULL;

SELECT COUNT(*) FROM PASSENGERS WHERE LASTNAME IS NULL;
```

COUNT(*)
0

COUNT(*)
0

COUNT(*)
0

COUNT(*)
0

COUNT(*)
0

COUNT(*)
0

Remove negative ages

```
DELETE FROM PASSENGERS
WHERE AGE<0;
```

Check Age Again

```
5 rows deleted.
```

NEG_AGE
0

4. Membershiptype Entity

There are in total 4 types of membership: bronze, silver, gold and royal

```
SELECT * FROM MEMBERSHIPTYPE;
```

```
ME MEMBERSHIPNAME       MEMBERSHIPFEE     PERIOD
-- -------------------- ------------- ----------
M1 bronze                         399          1
M2 silver                         599          1
M3 Gold                           799          1
M4 Royal                          999          1
```

5. Membershipjoinrecord Entity

Have a look at the number of records in the entity

```
SELECT COUNT(* ) FROM MEMBERSHIPJOINRECORDS;
```

```
  COUNT(*)
----------
     14634
```

Check the referential integrity for promotion, all valid.

```
SELECT * FROM MEMBERSHIPJOINRECORDS MR FULL OUTER JOIN PROMOTION P ON
MR.PROMOTION = P.PROMOTIONID WHERE P.PROMOTIONID IS NULL AND
MR.PROMOTION IS NOT NULL;
```

Check the referential integrity for membershiptype, all valid.

```
SELECT * FROM MEMBERSHIPJOINRECORDS MR FULL OUTER JOIN MEMBERSHIPTYPE MT
ON MR.MEMBERSHIPTYPEID = MT.MEMBERSHIPTYPEID WHERE MT.MEMBERSHIPTYPEID IS
NULL AND MR.MEMBERSHIPTYPEID IS NOT NULL;
```

Check the referential integrity for passengers, all valid.

```
SELECT * FROM MEMBERSHIPJOINRECORDS MR FULL OUTER JOIN PASSENGERS P ON
MR.PASSID = P.PASSID WHERE P.PASSID IS NULL;
```

```
no rows selected
no rows selected
no rows selected
```

Check the date range, it is valid, and all the date is within the range

```
SELECT COUNT(*) FROM MEMBERSHIPJOINRECORDS WHERE TO_CHAR(JOINDATE,'YY') >14
OR TO_CHAR(JOINDATE,'YY') <5;
```

```
  COUNT(*)
----------
         0
```

Check duplicate, we find there are no duplicates in membershipjoinrecords entity.

```
SELECT PASSID,JOINDATE,COUNT(*)  FROM MEMBERSHIPJOINRECORDS
GROUP BY PASSID,JOINDATE
HAVING COUNT(*)>1;
```

Check end date later than join date, the data is correct

```
SELECT
PASSID,TO_CHAR(JOINDATE,'YYYYMMDD'),TO_CHAR(ENDDATE,'YYYYMMDD'),COUNT(*)
FROM MEMBERSHIPJOINRECORDS
WHERE TO_CHAR(JOINDATE,'YYYYMMDD') > TO_CHAR(ENDDATE,'YYYYMMDD')
GROUP BY PASSID,TO_CHAR(JOINDATE,'YYYYMMDD'),TO_CHAR(ENDDATE,'YYYYMMDD');
```

```
no rows selected
no rows selected
```

6.Promotion Entity

There are 5 types of promotion and each promotion has its start date and end date, which may be related to our temporal entity. Each promotion owns its discount rate.

SELECT * FROM PROMOTION;

```
PROMOTIONI   DISCOUNT STARTDATE ENDDATE
---------- ---------- --------- ---------
P1                .1 01/JAN/05 31/AUG/05
P2               .15 01/NOV/06 31/MAY/07
P3               .17 01/MAR/08 30/APR/08
P4               .17 01/MAR/12 30/APR/12
P5                .2 01/JAN/13 30/APR/13
```

7. Aircraft Entity

Have a view on aircrafts , there is one is l/m, 3 empty

SELECT * FROM AIRCRAFTS FETCH FIRST 10 ROWS ONLY;

```
IATA ICAOCO MANUFACTURER                                      MODEL                                              WAK
---- ------ ------------------------------------------------- -------------------------------------------------- ---
100  F100   Fokker                                           100                                                M
141  B461   BAe                                               146-100 Pax                                        M
142  B462   BAe                                               146-200 Pax                                        M
143  B463   BAe                                               146-300 Pax                                        M
146         BAe                                               146 all pax models                                 M
14F         BAe                                               146 Freighter (-100/200/300QT & QC)                M
14X  B461   BAe                                               146 Freighter (-100QT & QC)                        M
14Y  B462   BAe                                               146 Freighter (-200QT & QC)                        M
14Z  B463   BAe                                               146 Freighter (-200QT & QC)                        M
310  A310   Airbus                                            A310 all pax models                                H

10 rows selected.
```

SELECT * FROM AIRCRAFTS WHERE WAKECATEGORY NOT IN ('H','M','L');

```
IATA ICAOCO MANUFACTURER                                      MODEL                                              WAK
---- ------ ------------------------------------------------- -------------------------------------------------- ---
BUS         Bus
JST         British Aerospace                                 Jetstream 31 / 32 / 41                             L/M
NDH  S65C   Eurocopter (Aerospatiale)                         SA365C / SA365N Dauphin 2
RFS         Road Feeder Service                               - Cargo Truck
```

find length! =3 and there are 5 same record for different IATACode.

SELECT * FROM AIRCRAFTS WHERE LENGTH(TRIM(IATACODE)) != 3;

SELECT COUNT(IATACODE), ICAOCODE, MANUFACTURER, MODEL, WAKECATEGORY
FROM AIRCRAFTS
GROUP BY ICAOCODE, MANUFACTURER, MODEL, WAKECATEGORY

```
IATA ICAOCO MANUFACTURER                                      MODEL                                              WAK
---- ------ ------------------------------------------------- -------------------------------------------------- ---
99   F100   Fokker                                           100                                                M
98   F100   Fokker                                           100                                                M
97   F100   Fokker                                           100                                                M
96   F100   Fokker                                           100                                                M
95   F100   Fokker                                           100                                                M
```

There are 6 records duplicates, and remove 5 ones that IATACode length less than 3

SELECT * FROM AIRCRAFTS WHERE ICAOCODE = 'F100' AND MANUFACTURER = 'FOKKER' AND MODEL = '100' AND WAKECATEGORY = 'M'

```
IATA ICAOCO MANUFACTURER                                      MODEL                                              WAK
---- ------ ------------------------------------------------- -------------------------------------------------- ---
100  F100   Fokker                                           100                                                M
99   F100   Fokker                                           100                                                M
98   F100   Fokker                                           100                                                M
97   F100   Fokker                                           100                                                M
96   F100   Fokker                                           100                                                M
95   F100   Fokker                                           100                                                M
```

Check they are not in flight table, no referential integrity violations.

```
SELECT * FROM FLIGHTS WHERE AIRCRAFTID IN (SELECT IATACODE FROM AIRCRAFTS
WHERE LENGTH(TRIM(IATACODE)) != 3);
```

```
no rows selected
```

Delete them from the aircraft entity

```
DELETE FROM AIRCRAFTS WHERE LENGTH(TRIM(IATACODE))! = 3;
```

```
5 rows deleted.
```

After removing records, only one record for IATACODE=F100 MANUFACTURER=Fokker MODEL=100 AND WAKECAEGORY=M

IATA	ICAOCO	MANUFACTURER	MODEL	WAK
100	F100	Fokker	100	M

8. Airline Entity

Find duplicates rows, the airlines with airlines 18723 and 18724 have same record of information , the airlines with airlines 168 and 169 with same record of information.

```
SELECT COUNT(AIRLINEID),LISTAGG(AIRLINEID,'_') WITHIN GROUP (ORDER BY AIRLINEID)AS
AIRLINES,NAME, ALIAS, IATACODE,ICAO,CALLSIGN,COUNTRY,ACTIVE
FROM AIRLINES
GROUP BY NAME, ALIAS, IATACODE, ICAO, CALLSIGN, COUNTRY, ACTIVE
HAVING COUNT(AIRLINEID) >1;
```

	COUNT(AIRLINEID)	AIRLINES	NAME	ALIAS	IATACODE	ICAO	CALLSIGN	COUNTRY	ACTIVE
1	2	18723_18724	WestAir Airlines	(null)	OE	(null)	(null)	United States	N
2	2	168_169	Angola Air Charter	(null)	(null)	AGO	ANGOLA CHARTER	Angola	N

Then double check the records in route entity. None of in route entity.

```
SELECT * FROM ROUTES WHERE AIRLINEID IN (18723,18724,168,169);
```

```
no rows selected
```

But check the records in provides, all of them are in the provides entity, so do not delete the records.

```
SELECT * FROM PROVIDES WHERE AIRLINEID IN (18723,18724,168,169);
```

```
 AIRLINEID  SERVICEID
---------- ----------
       168          3
       168          5
       168          6
       168          9
       169          1
       169          4
       169         11
     18723          3
     18723          5
     18723          6
     18723          9

 AIRLINEID  SERVICEID
---------- ----------
     18724          2
     18724          5
     18724          6
     18724          8

15 rows selected.
```

we found there is a row with AirlineId = -1, that need to be removed.

```
SELECT * FROM AIRLINES WHERE AIRLINEID ='-1';
```

	AIRLINEID	NAME	ALIAS	IATACODE	ICAO	CALLSIGN	COUNTRY	ACTIVE
1	-1	Unknown	(null)	-	N/A	(null)	Unknown	Y

```
DELETE FROM AIRLINES WHERE AIRLINEID =-1;
```

```
no rows selected
```

There are 5986 records left after cleaning.

```
SELECT COUNT(*) FROM AIRLINES;
```

```
  COUNT(*)
----------
      5986
```

9. Provides Entity

none of entity with primary key has null values

```
SELECT P.AIRLINEID,P.SERVICEID,A.AIRLINEID, S.SERVICEID
FROM PROVIDES P FULL OUTER JOIN AIRLINES A ON P.AIRLINEID = A.AIRLINEID FULL OUTER
JOIN AIRLINE_SERVICES S ON P.SERVICEID = S.SERVICEID
WHERE A.AIRLINEID IS NULL OR S.SERVICEID IS NULL;
```

```
no rows selected
```

no duplicates in provides

```
SELECT COUNT(DISTINCT AIRLINEID), AIRLINEID ,LISTAGG(AIRLINEID,'_') WITHIN
GROUP( ORDER BY AIRLINEID) AS AIRLINEID FROM PROVIDES
GROUP BY AIRLINEID
HAVING COUNT(DISTINCT AIRLINEID)>1;
```

```
no rows selected
```

10.Airline Service Entity

There are no duplicates within airline service entity

```
SELECT COUNT(SERVICEID), NAME, DESCRIPTION FROM AIRLINE_SERVICES
GROUP BY NAME, DESCRIPTION
HAVING COUNT(SERVICEID) > 1;
```

```
no rows selected
```

11.Routes Entity

check referential integrity, there is one record matched.

```
SELECT * FROM ROUTES R FULL OUTER JOIN AIRLINES AL ON R. AIRLINEID = AL.AIRLINEID
WHERE AL. AIRLINEID IS NULL
```

	ROUTEID	AIRLINEID	SOURCEAIRPORTID	DESTAIRPORTID	STOPS	EQUIPMENT	DISTANCE	SERVICECOST	NEWLYOPENED	AIRLINEID_1	NAME	ALIAS	IATACODE	ICAO	CALLSIGN	COUNTRY	ACTIVE
1	60000	9999	4029	2990	1	CR2	1234.23	24684.6	Y	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)

There are no duplicates records in routes entity excluding routeid

```
SELECT COUNT(ROUTEID), AIRLINEID, SOURCEAIRPORTID, DESTAIRPORTID, STOPS,
EQUIPMENT, DISTANCE, NEWLYOPENED
FROM ROUTES GROUP BY AIRLINEID, SOURCEAIRPORTID, DESTAIRPORTID, STOPS,
EQUIPMENT, DISTANCE, NEWLYOPENED
HAVING COUNT(ROUTEID)>1;
```

```
no rows selected
```

Delete the record with invalid AirportId , one row deleted.

```
DELETE FROM ROUTES WHERE ROUTEID =60000;
```

Check after removing

SELECT * FROM ROUTES WHERE ROUTEID = 60000;

```
no rows selected
```

Check routes between two airports, and there is no null primary key

SELECT AP1.AIRPORTID AS AP1_AIRPORTID, R.DESTAIRPORTID, R.SOURCEAIRPORTID,R.ROUTEID
FROM ROUTES R FULL OUTER JOIN AIRPORTS AP1 ON R.SOURCEAIRPORTID = AP1.AIRPORTID WHERE AP1.AIRPORTID IS NULL;

```
no rows selected
```

SELECT AP2.AIRPORTID AS AP2_AIRPORTID, R.DESTAIRPORTID, R.SOURCEAIRPORTID,R.ROUTEID
FROM ROUTES R FULL OUTER JOIN AIRPORTS AP2 ON R.DESTAIRPORTID = AP2.AIRPORTID WHERE AP2.AIRPORTID IS NULL;

```
no rows selected
```

12.Airports Entity

Number of airports information

SELECT COUNT(*) FROM AIRPORTS;

```
  COUNT(*)
----------
      7738
```

IATA_FAA 3-letter, ICAO 4, DST

SELECT * FROM AIRPORTS WHERE LENGTH(TRIM(IATA_FAACODE)) != 3;

```
no rows selected
```

There are many rows shown, and we will not delete the rows.

SELECT * FROM AIRPORTS WHERE LENGTH(TRIM(ICAO)) != 4;

	AIRPORTID	NAME	CITY	COUNTRY	IATA_FAACODE	ICAO	LATITUDE	LONGITUDE	ALTITUDE	TIMEZONE	DST
1	5829	Wotje Atoll Airport	Wotje Atoll	Marshall Islands	WTE	N36	9.46667	170.233	4	12	U
2	5827	Jaluit Airport	Jabor Jaluit Atoll	Marshall Islands	UIT	N55	5.90924	169.637	4	12	U
3	5825	Namorik Atoll Airport	Namorik Atoll	Marshall Islands	NDK	3N0	5.63167	168.125	15	12	U
4	5823	Mejit Atoll Airport	Mejit Atoll	Marshall Islands	MJB	Q30	10.2833	170.883	5	12	U
5	5814	Utirik Airport	Utirik Island	Marshall Islands	UTK	03N	11.222	169.852	4	12	U
6	5743	Kili Airport	Kili Island	Marshall Islands	KIO	Q51	5.64452	169.12	5	12	U
7	6126	Aleknagik Airport	Aleknagik	United States	WKK	5A8	59.2826	-158.618	66	-9	A
8	6127	Brookings Regional Airport	Brookings	United States	BKX	BKX	44.3048	-96.8169	1648	-6	A
9	6128	Mercer County Airport	Bluefield	United States	BLF	BLF	37.2958	-81.2077	2857	-5	A
10	6129	Kearney Municipal Airport	Kearney	United States	EAR	EAR	40.727	-99.0068	2131	-6	A
11	6130	Mid Delta Regional Airport	Greenville	United States	GLH	GLH	33.4829	-90.9856	131	-6	A
12	6131	Laughlin-Bullhead Intl	Bullhead	United States	IFP	IFP	35.1574	-114.56	695	-7	A
13	6132	Kingman Airport	Kingman	United States	IGM	IGM	35.2595	-113.938	3449	-7	A
14	6133	Tri Cities Airport	Pasco	United States	PSC	PSC	46.2647	-119.119	410	-8	A
15	6134	Akutan Seaplane Base	Akutan	United States	KQA	KQA	54.1325	-165.785	0	-9	A
16	6135	Grant County Airport	Silver City	United States	SVC	SVC	32.6365	-108.156	5446	-7	A
17	6136	Lopez Island Airport	Lopez	United States	LPS	S31	48.4839	-122.938	209	-8	A

Check AirportId using

SELECT * FROM AIRPORTS WHERE AIRPORTID <0;

	AIRPORTID	NAME	CITY	COUNTRY	IATA_FAACODE	ICAO	LATITUDE	LONGITUDE	ALTITUDE	TIMEZONE	DST
1	-5	Goroka	Goroka	Papua New Guinea	GKA	AYGA	-6.081689	145.391881	5282	10	U
2	-4	Goroka	Goroka	Papua New Guinea	GKA	AYGA	-6.081689	145.391881	5282	10	U
3	-3	Goroka	Goroka	Papua New Guinea	GKA	AYGA	-6.081689	145.391881	5282	10	U
4	-2	Goroka	Goroka	Papua New Guinea	GKA	AYGA	-6.081689	145.391881	5282	10	U
5	-1	Goroka	Goroka	Papua New Guinea	GKA	AYGA	-6.081689	145.391881	5282	10	U

Remove invalid AirportId

DELETE FROM AIRPORTS WHERE AIRPORTID <0;

Look at the data after cleaning

SELECT * FROM AIRPORTS WHERE AIRPORTID <0;

```
5 rows deleted.


NEG_AIRPORTID
-------------
            0
```

We have found that many of the cities with same names may locate in different country with different daylight-saving time but assume they are correct and not change it.

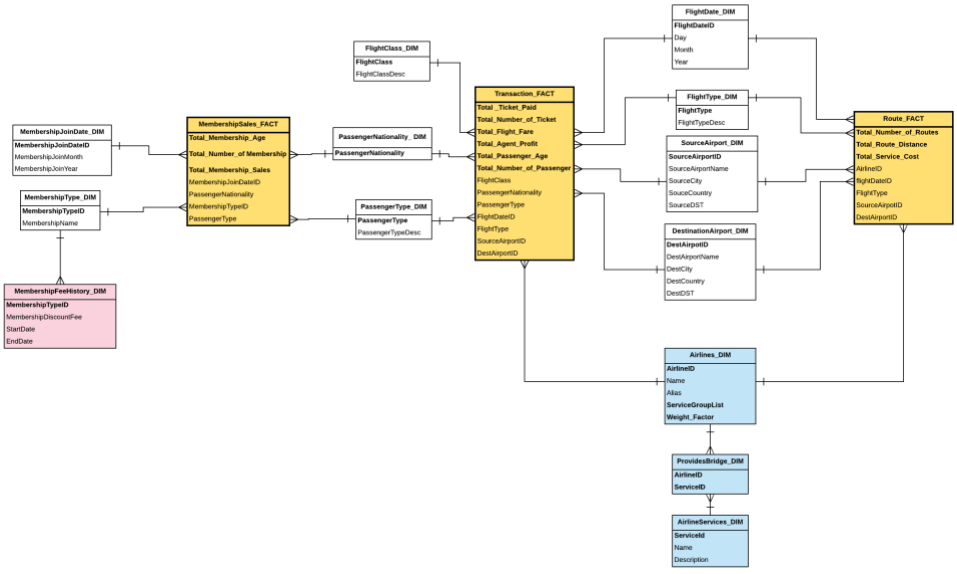
```
SELECT COUNT(DISTINCT COUNTRY),CITY,DST,LISTAGG(COUNTRY,'_')WITHIN GROUP(ORDER BY COUNTRY) FROM AIRPORTS
GROUP BY CITY,DST
HAVING COUNT(DISTINCT COUNTRY)>1;
```

All Rows Fetched: 36 in 0.404 seconds

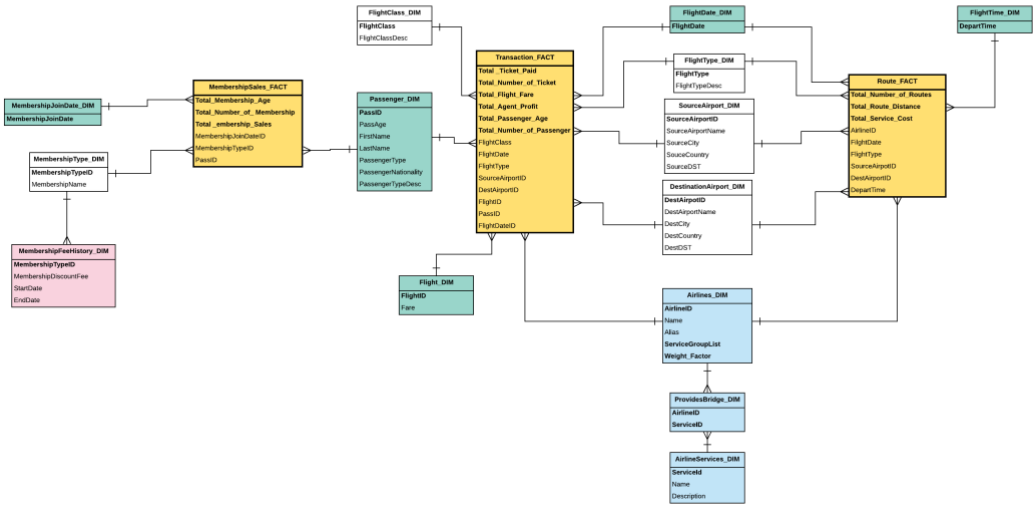
	COUNT(DISTINCTCOUNTRY)	CITY	DST	LISTAGG(COUNTRY,'_')WITHINGROUP(ORDERBYCOUNTRY)
1	2	Faro	E	Portugal_Sweden
2	2	Oran	N	Algeria_Algeria_Argentina
3	2	Brest	E	Belarus_France
4	2	Pemba	U	Mozambique_Tanzania
5	2	London	A	Canada_United States
6	2	Trenton	A	Canada_United States_United States_United States
7	2	Hamilton	A	Canada_United States
8	2	Marathon	A	Canada_United States
9	2	Samjiyon	U	North Korea_South Korea
10	2	San Jose	U	Costa Rica_Costa Rica_Guatemala
11	2	Santiago	U	Dominican Republic_Panama
12	2	Trinidad	U	Bolivia_Cuba
13	2	Trujillo	U	Honduras_Peru
14	2	Victoria	A	Canada_Canada_Canada_United States
15	2	Waterloo	A	Canada_United States
16	2	Greenwood	A	Canada_United States
17	2	Hyderabad	N	India_Pakistan
18	2	Newcastle	E	Ireland_United Kingdom

c. Two versions of star/snowflake schema diagrams.

version 1



version 2



d. Explanation of why you chose hierarchy

In these two star schemas, we use **non-hierarchy** to complete them. Hierarchy is often used to show the level of granularity and starts from most detailed into more general. But in our star schema, we don't need to change any dimension from most detailed one into more general ones because we only need most detailed dimensions and we don't separate them into detailed one and general one. Therefore, we choose to use non-hierarchy.

e. Reasons of the choose SCD type 4 for temporal dimension

In temporal dimension, we include the price of membership fee. And then we use history fee to calculate total membership sales.

If we use SCD type 2, it will put history fee into MembershipType dimension. After that, we will find there are many rows with different history fee but same values in other columns. This will cause redundancy.

If we use SCD type 6, it will include current fee and previous fee at the same time. But in our star schema, we don't need previous fee to calculate total membership sales. Some columns will be useless.

So, we choose to use **SCD type 4** to create a new dimension to include all values of membership fee. And then we can use all membership fee to calculate total membership sales.

f. Explanation of the difference among the two versions

The main difference between two versions will be their level of aggregation. Version 1 is in level 2, which means high level of aggregation but version 2 is in level 0, which means no aggregation. There are two difference among two versions.

1. When we observe the aggregate attribute in fact tables, we will find those values of counting attributes are always be 1 because there is no aggregation in version 2.

2. Also, we find the dimensions between two versions are also different. Version 2 drill down of the FlightDate and MembershipJoinDate dimensions. And add Passenger, Flight and FlightTime dimensions to discard any grouping dimension attributes.

Task C.2(star schemas)

a&c. SQL statements of Version 1 and screen shots of tables created.

There are three steps to create version 1 (19 tables in total):
1. create new entity of temporal table (MemberhipFeeHistory table, 1 table)
2. create all dimension tables (12 tables in total)
3. create all fact tables (including temp fact table, 6 tables in total)

1. create new entity of MembershipFeeHistory (temporal table)

```
CREATE TABLE membershipfeehistory
    AS
        SELECT DISTINCT
            mt.membershiptypeid,
            mt.membershipfee * ( 1 - p.discount ) AS membershipdiscountfee,
            p.startdate,
            p.enddate
        FROM
            membershipjoinrecords m,
            promotion p,
            membershiptype mt
        WHERE
            m.promotion = p.promotionid
            AND   m.membershiptypeid = mt.membershiptypeid;
```

	MEMBERSHIPTYPEID	MEMBERSHIPDISCOUNTFEE	STARTDATE	ENDDATE
1	M4	899.1	01/JAN/05	31/AUG/05
2	M3	719.1	01/JAN/05	31/AUG/05
3	M2	479.2	01/JAN/13	30/APR/13
4	M3	663.17	01/MAR/12	30/APR/12
5	M1	339.15	01/NOV/06	31/MAY/07
6	M4	799.2	01/JAN/13	30/APR/13
7	M3	639.2	01/JAN/13	30/APR/13
8	M1	331.17	01/MAR/08	30/APR/08
9	M4	849.15	01/NOV/06	31/MAY/07
10	M1	331.17	01/MAR/12	30/APR/12
11	M1	359.1	01/JAN/05	31/AUG/05
12	M2	539.1	01/JAN/05	31/AUG/05
13	M2	497.17	01/MAR/08	30/APR/08
14	M2	497.17	01/MAR/12	30/APR/12
15	M3	679.15	01/NOV/06	31/MAY/07
16	M4	829.17	01/MAR/12	30/APR/12
17	M4	829.17	01/MAR/08	30/APR/08
18	M1	319.2	01/JAN/13	30/APR/13
19	M2	509.15	01/NOV/06	31/MAY/07
20	M3	663.17	01/MAR/08	30/APR/08
21	M1	399	01/SEP/05	31/OCT/06
22	M2	599	01/SEP/05	31/OCT/06
23	M3	799	01/SEP/05	31/OCT/06
24	M4	999	01/SEP/05	31/OCT/06
25	M1	399	01/JUN/07	29/FEB/08
26	M2	599	01/JUN/07	29/FEB/08
27	M3	799	01/JUN/07	29/FEB/08
28	M4	999	01/JUN/07	29/FEB/08
29	M1	399	01/MAY/08	29/FEB/12
30	M2	599	01/MAY/08	29/FEB/12
31	M3	799	01/MAY/08	29/FEB/12
32	M4	999	01/MAY/08	29/FEB/12
33	M1	399	01/MAY/12	31/DEC/12
34	M2	599	01/MAY/12	31/DEC/12
35	M3	799	01/MAY/12	31/DEC/12
36	M4	999	01/MAY/12	31/DEC/12
37	M1	399	01/MAY/13	31/DEC/14
38	M2	599	01/MAY/13	31/DEC/14
39	M3	799	01/MAY/13	31/DEC/14
40	M4	999	01/MAY/13	31/DEC/14

(all data)

2. membershipjoindate_dim_v1

```
CREATE TABLE membershipjoindate_dim_v1
    AS
        SELECT DISTINCT
            TO_CHAR(joindate,'yyyymm') AS membershipjoindateid,
            TO_CHAR(joindate,'mm') AS membershipjoinmonth,
            TO_CHAR(joindate,'yyyy') AS membershipjoinyear
        FROM
            membershipjoinrecords;
```

	MEMBERSHIPJOINDATEID	MEMBERSHIPJOINMONTH	MEMBERSHIPJOINYEAR
1	200610	10	2006
2	200809	09	2008
3	200607	07	2006
4	200705	05	2007
5	201305	05	2013
6	200711	11	2007
7	200606	06	2006
8	201201	01	2012
9	201306	06	2013
10	201110	10	2011
11	201202	02	2012
12	201303	03	2013
13	201205	05	2012
14	200810	10	2008
15	200612	12	2006
16	200801	01	2008
17	201011	11	2010
18	201112	12	2011
19	201207	07	2012
20	201105	05	2011
21	200512	12	2005
22	201111	11	2011
23	201203	03	2012
24	201310	10	2013
25	200611	11	2006

(part of data)

3. membershiptype_dim_v1

```
CREATE TABLE membershiptype_dim_v1
    AS
        SELECT DISTINCT
            membershiptypeid,
            membershipname
        FROM
            membershiptype;
```

	MEMBERSHIPTYPEID	MEMBERSHIPNAME
1	M2	silver
2	M3	Gold
3	M4	Royal
4	M1	bronze

(all data)

4. membershipfeehistory_dim_v1

```
CREATE TABLE membershipfeehistory_dim_v1
    AS
        SELECT
            *
        FROM
            membershipfeehistory;
```

	MEMBERSHIPTYPEID	MEMBERSHIPDISCOUNTFEE	STARTDATE	ENDDATE
1	M4	899.1	01/JAN/05	31/AUG/05
2	M3	719.1	01/JAN/05	31/AUG/05
3	M2	479.2	01/JAN/13	30/APR/13
4	M3	663.17	01/MAR/12	30/APR/12
5	M1	339.15	01/NOV/06	31/MAY/07
6	M4	799.2	01/JAN/13	30/APR/13
7	M3	639.2	01/JAN/13	30/APR/13
8	M1	331.17	01/MAR/08	30/APR/08
9	M4	849.15	01/NOV/06	31/MAY/07
10	M1	331.17	01/MAR/12	30/APR/12
11	M1	359.1	01/JAN/05	31/AUG/05
12	M2	539.1	01/JAN/05	31/AUG/05
13	M2	497.17	01/MAR/08	30/APR/08
14	M2	497.17	01/MAR/12	30/APR/12
15	M3	679.15	01/NOV/06	31/MAY/07
16	M4	829.17	01/MAR/12	30/APR/12
17	M4	829.17	01/MAR/08	30/APR/08
18	M1	319.2	01/JAN/13	30/APR/13
19	M2	509.15	01/NOV/06	31/MAY/07
20	M3	663.17	01/MAR/08	30/APR/08
21	M1	399	01/SEP/05	31/OCT/06
22	M2	599	01/SEP/05	31/OCT/06
23	M3	799	01/SEP/05	31/OCT/06
24	M4	999	01/SEP/05	31/OCT/06
25	M1	399	01/JUN/07	29/FEB/08
26	M2	599	01/JUN/07	29/FEB/08
27	M3	799	01/JUN/07	29/FEB/08
28	M4	999	01/JUN/07	29/FEB/08
29	M1	399	01/MAY/08	29/FEB/12
30	M2	599	01/MAY/08	29/FEB/12
31	M3	799	01/MAY/08	29/FEB/12
32	M4	999	01/MAY/08	29/FEB/12
33	M1	399	01/MAY/12	31/DEC/12
34	M2	599	01/MAY/12	31/DEC/12
35	M3	799	01/MAY/12	31/DEC/12
36	M4	999	01/MAY/12	31/DEC/12
37	M1	399	01/MAY/13	31/DEC/14
38	M2	599	01/MAY/13	31/DEC/14
39	M3	799	01/MAY/13	31/DEC/14
40	M4	999	01/MAY/13	31/DEC/14

(all data)

5. passengernationality_dim_v1

```
CREATE TABLE passengernationality_dim_v1
    AS
        SELECT DISTINCT
            nationality
        FROM
            passengers;
```

	NATIONALITY
1	Mauritian
2	Liberian
3	Belizean
4	Bahamian
5	Saudi
6	San Marinese
7	Iranian
8	Hungarian
9	Ivorian
10	Filipino
11	Northern Irish
12	Ecuadorean
13	Taiwanese
14	Israeli
15	Kazakhstani
16	Emirian
17	Marshallese
18	Nigerien
19	Egyptian
20	Italian

(part of data)

6. passengertype_dim_v1

```
CREATE TABLE passengertype_dim_v1 (
    passengertype        NUMBER,
    passengertypedesc    VARCHAR(20)
);

INSERT INTO passengertype_dim_v1 VALUES (
    1,
    'Children'
);

INSERT INTO passengertype_dim_v1 VALUES (
    2,
    'Teenager'
);

INSERT INTO passengertype_dim_v1 VALUES (
    3,
    'Adult'
);

INSERT INTO passengertype_dim_v1 VALUES (
    4,
    'Elder'
);
```

	PASSENGERTYPE	PASSENGERTYPEDESC
1	1	Children
2	2	Teenager
3	3	Adult
4	4	Elder

(all data)

7. flightclass_dim_v1

```
CREATE TABLE flightclass_dim_v1 (
    flightclass       NUMBER,
    flightclassdesc   VARCHAR(20)
);

INSERT INTO flightclass_dim_v1 VALUES (
    1,
    'First Class'
);

INSERT INTO flightclass_dim_v1 VALUES (
    2,
    'Business Class'
);

INSERT INTO flightclass_dim_v1 VALUES (
    3,
    'Economy Class'
);
```

	FLIGHTCLASS	FLIGHTCLASSDESC
1	1	First Class
2	2	Business Class
3	3	Economy Class

(all data)

8. flightdate_dim_v1

```
CREATE TABLE flightdate_dim_v1
    AS
        SELECT DISTINCT
            TO_CHAR(flightdate,'YYYYMMDD') AS flightdateid,
            TO_CHAR(flightdate,'DY') AS day,
            TO_CHAR(flightdate,'Mon') AS month,
            TO_CHAR(flightdate,'YYYY') AS year
        FROM
            flights;
```

	FLIGHTDATEID	DAY	MONTH	YEAR
1	20071029	MON	Oct	2007
2	20090326	THU	Mar	2009
3	20070220	TUE	Feb	2007
4	20071205	WED	Dec	2007
5	20080118	FRI	Jan	2008
6	20070309	FRI	Mar	2007
7	20070526	SAT	May	2007
8	20080902	TUE	Sep	2008
9	20080704	FRI	Jul	2008
10	20080903	WED	Sep	2008
11	20081117	MON	Nov	2008
12	20090824	MON	Aug	2009
13	20081121	FRI	Nov	2008
14	20070221	WED	Feb	2007
15	20070328	WED	Mar	2007

(part of data)

9. flighttype_dim_v1

```
CREATE TABLE flighttype_dim_v1 (
    flighttype       NUMBER,
    flighttypedesc   VARCHAR(20)
);

INSERT INTO flighttype_dim_v1 VALUES (
    1,
    'Domestic'
);

INSERT INTO flighttype_dim_v1 VALUES (
    2,
    'International'
);
```

	FLIGHTTYPE	FLIGHTTYPEDESC
1	1	Domestic
2	2	International

(all data)

10. We build one airport_dim_v1 to include information of Sourceairport_dim_v1 and Destairport_dim_v1

```
CREATE TABLE airport_dim_v1
    AS
        SELECT DISTINCT
            airportid,
            name,
            city,
            country,
            dst
        FROM
            airports;
```

	AIRPORTID	NAME	CITY	COUNTRY	DST
1	2	Madang	Madang	Papua New Guinea	U
2	10	Thule Air Base	Thule	Greenland	E
3	12	Egilsstadir	Egilsstadir	Iceland	N
4	40	Clyde River	Clyde River	Canada	A
5	64	Geraldton Greenstone Regional	Geraldton	Canada	A
6	67	Dryden Rgnl	Dryden	Canada	A
7	88	Mayo	Mayo	Canada	A
8	98	Cold Lake	Cold Lake	Canada	A
9	102	Peace River	Peace River	Canada	A
10	326	Rechlin Larz	Rechlin-laerz	Germany	E

(part of data)

11. airlines_dim_v1

```
CREATE TABLE airlines_dim_v1
    AS
        SELECT
            a.airlineid,
            a.name,
            a.alias,
            round(1 / COUNT(p.serviceid),2) AS weight_factor,
            LISTAGG(p.serviceid,
            '_') WITHIN GROUP(
            ORDER BY
                p.serviceid
            ) AS servicegrouplist
        FROM
            airlines a,
            provides p
        WHERE
            a.airlineid = p.airlineid
        GROUP BY
            a.airlineid,
            a.name,
            a.alias;
```

	AIRLINEID	NAME	ALIAS	WEIGHT_FACTOR	SERVICEGROUPLIST
1	1	Private flight	(null)	0.33	1_4_10
2	2	135 Airways	(null)	0.25	1_4_7_8
3	3	1Time Airline	(null)	0.33	3_5_10
4	4	2 Sqn No 1 Elementary Flying Training School	(null)	0.25	3_5_6_9
5	5	213 Flight Unit	(null)	0.25	3_4_6_9
6	6	223 Flight Unit State Airline	(null)	0.25	1_4_7_8
7	7	224th Flight Unit	(null)	0.25	2_5_6_8
8	8	247 Jet Ltd	(null)	0.33	3_4_11
9	9	3D Aviation	(null)	0.33	1_5_10
10	10	40-Mile Air	(null)	0.33	3_5_10
11	11	4D Air	(null)	0.25	2_4_6_9
12	12	611897 Alberta Limited	(null)	0.33	1_4_11
13	13	Ansett Australia	(null)	0.33	3_4_11
14	14	Abacus International	(null)	0.33	3_5_10
15	15	Abelag Aviation	(null)	0.25	3_4_6_9

(part of data)

12. providesbridge_dim_v1

```
CREATE TABLE providesbridge_dim_v1
    AS
        SELECT
            *
        FROM
            provides;
```

	AIRLINEID	SERVICEID
1	356	1
2	356	4
3	356	6
4	356	9
5	357	3
6	357	4
7	357	11
8	358	2
9	358	4
10	358	11
11	359	2
12	359	5
13	359	11
14	360	2
15	360	5

(part of data)

13. airlineservices_dim_v1

```
CREATE TABLE airlineservices_dim_v1
    AS
        SELECT
            *
        FROM
            airline_services;
```

	SERVICEID	NAME	DESCRIPTION
1	1	Extra weight 20	Customer can buy up to 20kg extra luggage weight
2	2	Extra weight 10	Customer can buy up to 10kg extra luggage weight
3	3	Extra weight 5	Customer can buy up to 5kg extra luggage weight
4	4	In-flight breakfast	breakfast served to passengers on board
5	5	In-flight meal	meal served to passengers on board
6	6	In-flight fast food	fast food served to passengers on board
7	7	In-flight beverage	drink served to passengers on board
8	8	In-flight movies	Customer can view online movies
9	9	In-flight music	Customer can listen to music
10	10	In-flight games	Customer can play games
11	11	In-flight internet	Customer can access internet via Wifi on board

(all data)

14. Temp_membershipsales_fact_v1 (for MembershipSales fact table)

```sql
--1. MembershipSales_FACT
--1.1 create Temp Fact table

CREATE TABLE temp_membershipsales_fact_v1
    AS
        SELECT
            to_char(m.joindate,'yyyymm') AS membershipjoindateid,
            p.nationality,
            p.age,
            m.membershiptypeid,
            mh.membershipdiscountfee,
            p.passid
        FROM
            passengers p,
            membershipjoinrecords m,
            membershipfeehistory mh
        WHERE
            p.passid = m.passid
            AND   m.membershiptypeid = mh.membershiptypeid
                  AND   m.joindate BETWEEN mh.startdate AND mh.enddate;
--1.2 create passengertype attribute

ALTER TABLE temp_membershipsales_fact_v1 ADD (
    passengertype   VARCHAR(20)
);

UPDATE temp_membershipsales_fact_v1
    SET
        passengertype = 1
WHERE
    age < 11;

UPDATE temp_membershipsales_fact_v1
    SET
        passengertype = 2
WHERE
    age >= 11
    AND   age <= 17;

UPDATE temp_membershipsales_fact_v1
    SET
        passengertype = 3
WHERE
    age >= 18
    AND   age <= 60;

UPDATE temp_membershipsales_fact_v1
    SET
        passengertype = 4
WHERE
    age > 60;

COMMIT;

SELECT
    COUNT(*)
FROM
    temp_membershipsales_fact_v1
WHERE
    passengertype IS NULL; --check if all passengertype is already changed
```

	MEMBERSHIPJOINDATEID	NATIONALITY	AGE	MEMBERSHIPTYPEID	MEMBERSHIPDISCOUNTFEE	PASSID	PASSENGERTYPE
1	201312	Australian	81	M4	999	5648	4
2	201410	Guinean	38	M2	599	5658	3
3	201403	Somali	23	M3	799	1973	3
4	200608	Dominican	55	M1	399	5927	3
5	201103	Australian	0	M3	799	5014	1
6	201303	Cape Verdean	6	M4	799.2	6725	1
7	200507	Fijian	18	M4	899.1	4901	3
8	200708	Australian	43	M2	599	4548	3
9	201101	Australian	69	M1	399	3456	4
10	200903	Nicaraguan	6	M4	999	6245	1

(part of data)

15. Membershipsales_fact_v1

```sql
--1.3 create membershipsales_fact table for version 1(level 2)

CREATE TABLE membershipsales_fact_v1
    AS
        SELECT
            membershipjoindateid,
            passengertype,
            membershiptypeid,
            SUM(membershipdiscountfee) AS total_membership_sales,
            COUNT(passid) AS total_number_of_membership,
            SUM(age) AS total_membership_age
        FROM
            temp_membershipsales_fact_v1
        GROUP BY
            membershipjoindateid,
            passengertype,
            membershiptypeid;
```

	MEMBERSHIPJOINDATEID	PASSENGERTYPE	MEMBERSHIPTYPEID	TOTAL_MEMBERSHIP_SALES	TOTAL_NUMBER_OF_MEMBERSHIP	TOTAL_MEMBERSHIP_AGE
1	200507	3	M4	14385.6	16	563
2	201303	3	M2	5750.4	12	469
3	201108	3	M1	3990	10	428
4	201012	4	M4	7992	8	584
5	201005	1	M3	7191	9	41
6	200905	2	M2	2995	5	72
7	200809	3	M4	22977	23	903
8	200706	3	M3	7990	10	449
9	201411	1	M2	4193	7	19
10	200812	4	M3	10387	13	946
11	200708	4	M4	10989	11	763
12	201002	3	M4	15984	16	558
13	201010	3	M2	10183	17	615
14	201007	1	M4	5994	6	33
15	200502	3	M4	8991	10	370

(part of data)

16. Temp_transaction_fact_v1 (for Transaction fact table)

```
CREATE TABLE temp_transaction_fact_v1
        AS
                SELECT
                    t.totalpaid,
                    f.fare,
                    p.age,
                    p.nationality AS passengernationality,
                    to_char(f.flightdate,'yyyymmdd') AS flightdateid,
                    r.sourceairportid,
                    r.destairportid,
                    p.passid,
                    source_a.country AS source_country,
                    dest_a.country AS dest_country,
                    a.airlineid
                FROM
                        routes r,
                        airlines a,
                        flights f,
                        transactions t,
                        passengers p,
                        airports source_a,
                        airports dest_a
                    WHERE
            r.routeid = f.routeid
            AND   f.flightid = t.flightid
                AND   t.passid = p.passid
                    AND   r.sourceairportid = source_a.airportid
                        AND   r.destairportid = dest_a.airportid
                            AND a.airlineid = r.airlineid;
--2.2 create flighttype, flightclass and passengertype attributes

ALTER TABLE temp_transaction_fact_v1 ADD (
    flighttype      VARCHAR(20),
    flightclass     VARCHAR(20),
    passengertype   VARCHAR(20)
);

UPDATE temp_transaction_fact_v1
    SET
        flighttype = 1
WHERE
    source_country = dest_country;

UPDATE temp_transaction_fact_v1
    SET
        flighttype = 2
WHERE
    source_country != dest_country;

UPDATE temp_transaction_fact_v1
    SET
        flightclass = 1
WHERE
    totalpaid >= 2 * fare;

UPDATE temp_transaction_fact_v1
    SET
        flightclass = 2
WHERE
    totalpaid < 2 * fare
    AND   totalpaid >= 1.5 * fare;
```

```
UPDATE temp_transaction_fact_v1
    SET
        flightclass = 3
WHERE
    totalpaid < 1.5 * fare;

UPDATE temp_transaction_fact_v1
    SET
        passengertype = 1
WHERE
    age < 11;

UPDATE temp_transaction_fact_v1
    SET
        passengertype = 2
WHERE
    age >= 11
    AND   age <= 17;

UPDATE temp_transaction_fact_v1
    SET
        passengertype = 3
WHERE
    age >= 18
    AND   age <= 60;

UPDATE temp_transaction_fact_v1
    SET
        passengertype = 4
WHERE
    age > 60;

COMMIT;
```

	TOTALPAID	FARE	AGE	PASSENGERNATIONALITY	FLIGHTDATEID	SOURCEAIRPORTID	DESTAIRPORTID	PASSID	SOURCE_COUNTRY	DEST_COUNTRY	AIRLINEID	FLIGHTTYPE	FLIGHTCLA
1	203.48	33.05	14	Australian	20080714	3076	3043	2334	Bangladesh	India	1359	2	1
2	904.47	375.96	8	Australian	20080531	3076	3093	4587	Bangladesh	India	1359	2	1
3	226.54	141.48	85	Australian	20071021	108	156	5241	Canada	Canada	2692	1	2
4	316.62	229.03	35	Saudi	20071127	156	108	1814	Canada	Canada	2692	1	3
5	252.78	61.27	34	Beninese	20080106	3370	3368	6318	China	China	4026	1	1
6	151.89	79.91	5	Haitian	20080812	3393	3368	3699	China	China	4026	1	2
7	187.95	97.07	50	Swedish	20080427	3371	3379	6884	China	China	4026	1	2
8	467.09	238.7	77	Australian	20080310	3368	3393	6433	China	China	4026	1	2
9	670.22	258.82	6	Australian	20070726	3368	3393	5198	China	China	4026	1	1
10	295.9	89.38	11	Australian	20080719	6187	3205	3471	Vietnam	Vietnam	3850	1	1
11	234.3	157.49	40	Australian	20081202	3393	3382	2669	China	China	2942	1	3
12	482.91	219.47	17	Australian	20071111	3382	3406	6546	China	China	2942	1	1
13	314.47	61.14	79	Australian	20090818	3387	6361	8307	China	China	2942	1	1
14	253.25	221.93	40	Australian	20080425	3388	3382	1120	China	China	2942	1	3
15	359.76	76.18	51	Irish	20080205	3373	3382	7044	China	China	2942	1	1
16	356.3	283.14	48	Australian	20070612	3369	6345	9189	China	China	2942	1	3

(part of data)

17. ransaction_fact_v1

```
CREATE TABLE transaction_fact_v1
    AS
        SELECT
            flightclass,
            passengernationality,
            passengertype,
            flightdateid,
            flighttype,
            sourceairportid,
            destairportid,
            airlineid,
            SUM(totalpaid) AS total_ticket_paid,
            COUNT(*) AS total_number_of_ticket,
            SUM(fare) AS total_flight_fare,
            SUM(totalpaid - fare) AS total_agent_profit,
            SUM(age) AS total_passenger_age,
            COUNT(passid) AS total_number_of_passenger
        FROM
            temp_transaction_fact_v1
        GROUP BY
            flightclass,
            passengernationality,
            passengertype,
            flightdateid,
            flighttype,
            sourceairportid,
            destairportid,
            airlineid;
```

	FLIGHTCLASS	PASSENGERNATIONALITY	PASSENGERTYPE	FLIGHTDATEID	FLIGHTTYPE	SOURCEAIRPORTID	DESTAIRPORTID	AIRLINEID	TOTAL_TICKET_PAID	TOTAL_NUMBER_OF_TICKET	TOTAL_FLIGHT_FARE	TOTAL_FLIGHT_FARE	TOTAL_AGENT_PROFIT	TOTAL_PASSENGER_AGE	TOTAL_NUMBER_OF_PASSENGER
1	2	Swedish	3	20080427	1	3371	3379	4026	187.95	1	97.07	33.05	170.43	14	1
2	3	Australian	3	20081202	1	3393	3382	2942	234.3	1	157.49	89.38	206.52	11	1
3	2	Libyan	4	20070316	1	3484	3687	439	247.05	1	157.64	231.82	60.12	73	1
4	2	Australian	3	20070923	2	813	879	683	692.85	1	300.89	514.52	23.69	63	1
5	1	Australian	3	20080516	2	3320	2276	2091	1830.32	1	808.97	236.59	438.04	68	1
6	3	Australian	1	20080516	2	3320	2276	2091	910.12	1	808.97	808.97	67.04	37	1
7	3	Australian	4	20090428	2	2276	3320	2091	4118.74	5	3777.25	808.97	885.49	76	1
8	3	Nigerian	3	20090428	2	2276	3320	2091	806.88	1	755.45	808.97	109.75	70	1
9	2	Australian	1	20080629	1	3417	6712	55	232.3	1	136.63	755.45	107.29	8	1
10	1	Egyptian	4	20090616	1	4144	3379	15999	393.91	1	157.17	755.45	52.81	80	1
11	2	Australian	4	20080406	1	3399	3391	15999	519.67	1	277.92	176.55	351.7	66	1
12	3	Australian	1	20070404	2	1918	193	330	611.82	1	520.33	38.58	87.26	60	1
13	1	Australian	3	20090621	2	3361	156	330	6460.32	2	3062	276.38	98.39	14	1
14	2	Australian	1	20080713	2	3361	156	330	1990.22	1	1266.57	1531	83.6	48	1
15	3	Australian	3	20080713	2	3361	156	330	5294.3	4	5066.28	1531	0	13	1

(part of data)

18. Temp_route_fact_v1 (for Route fact table)

```
CREATE TABLE temp_route_fact_v1
    AS
        SELECT
            r.airlineid,
            to_char(f.flightdate,'yyyymmdd') AS flightdateid,
            r.sourceairportid,
            r.destairportid,
            r.routeid,
            r.distance,
            r.servicecost,
            source_a.country AS source_country,
            dest_a.country AS dest_country
        FROM
            flights f,
            routes r,
            airports source_a,
            airports dest_a
        WHERE
            f.routeid = r.routeid
            AND   r.sourceairportid = source_a.airportid
                AND   r.destairportid = dest_a.airportid;
--3.2 create flighttype attribute

ALTER TABLE temp_route_fact_v1 ADD (
    flighttype   VARCHAR(20)
);

UPDATE temp_route_fact_v1
    SET
        flighttype = 1
WHERE
    source_country = dest_country;

UPDATE temp_route_fact_v1
    SET
        flighttype = 2
WHERE
    source_country != dest_country;

COMMIT;

SELECT
    COUNT(*)
FROM
    temp_route_fact_v1
WHERE
    flighttype IS NULL; --check if all flighttype is already changed
```

	AIRLINEID	FLIGHTDATEID	SOURCEAIRPORTID	DESTAIRPORTID	ROUTEID	DISTANCE	SERVICECOST	SOURCE_COUNTRY	DEST_COUNTRY	FLIGHTTYPE
1	2421	20080914	547	535	12168	472.77	9455.4	United Kingdom	United Kingdom	1
2	2421	20081124	547	478	12169	249.61	4992.2	United Kingdom	United Kingdom	1
3	2421	20080417	547	478	12169	249.61	4992.2	United Kingdom	United Kingdom	1
4	2421	20090221	1386	495	12170	363.55	7271	France	United Kingdom	2
5	2421	20070130	1386	495	12170	363.55	7271	France	United Kingdom	2
6	2421	20090528	1386	495	12170	363.55	7271	France	United Kingdom	2
7	2421	20080303	1415	495	12171	321.47	6429.4	France	United Kingdom	2
8	2421	20070829	1415	495	12171	321.47	6429.4	France	United Kingdom	2
9	2421	20081010	1415	495	12171	321.47	6429.4	France	United Kingdom	2
10	2421	20090630	603	534	12172	457.22	9144.4	Ireland	United Kingdom	2
11	2421	20070418	495	1230	12173	1606.78	32135.6	United Kingdom	Spain	2
12	2421	20090103	495	1212	12174	1409.95	28199	United Kingdom	Spain	2
13	2421	20080802	495	1212	12174	1409.95	28199	United Kingdom	Spain	2
14	2421	20091009	495	580	12175	448.47	8969.4	United Kingdom	Netherlands	2
15	2421	20090207	495	535	12178	571.65	11433	United Kingdom	United Kingdom	1
16	2421	20080617	495	535	12178	571.65	11433	United Kingdom	United Kingdom	1
17	2421	20081002	495	535	12178	571.65	11433	United Kingdom	United Kingdom	1
18	2421	20070503	495	352	12182	778.32	15566.4	United Kingdom	Germany	2
19	2421	20071130	495	1418	12188	422.6	8452	United Kingdom	France	2
20	2421	20080213	495	1386	12189	363.55	7271	United Kingdom	France	2

(part of data)

19. Route_fact_v1

```
CREATE TABLE route_fact_v1
    AS
        SELECT
            airlineid,
            flightdateid,
            flighttype,
            sourceairportid,
            destairportid,
            COUNT(routeid) AS total_number_of_routes,
            SUM(distance) AS total_route_distance,
            SUM(servicecost) AS total_service_cost
        FROM
            temp_route_fact_v1
        GROUP BY
            airlineid,
            flightdateid,
            flighttype,
            sourceairportid,
            destairportid;
```

	AIRLINEID	FLIGHTDATEID	FLIGHTTYPE	SOURCEAIRPORTID	DESTAIRPORTID	TOTAL_NUMBER_OF_ROUTES	TOTAL_ROUTE_DISTANCE	TOTAL_SERVICE_COST
1	2421	20080303	2	1415	495	1	321.47	6429.4
2	2421	20070121	2	350	469	1	879.21	17584.2
3	2421	20091227	1	541	533	1	154.35	3087
4	1359	20080622	2	2179	1555	1	4339.04	86780.8
5	1359	20090829	2	3043	3076	1	239.15	4783
6	1359	20081217	1	3069	3076	1	228.85	4577
7	1359	20080203	2	3076	2179	1	3617.53	72350.6
8	1359	20080424	2	3076	3304	1	2642.21	52844.2
9	1359	20071021	2	3125	3076	1	661.75	13235
10	1359	20091224	2	2082	3076	1	4407.31	88146.2
11	4255	20091231	2	3272	3304	1	1486.58	29731.6
12	4255	20081020	2	3077	3272	1	1933.85	38677
13	4255	20080919	2	3077	3272	1	1933.85	38677
14	4255	20080103	2	3304	3272	1	1486.58	29731.6
15	4026	20080619	1	3371	3383	1	634.96	12699.2

(part of data)

b&c. SQL statements of Version 2 and screen shots of tables created.

There are two steps to create version 2 (19 tables):
1. create all dimension tables (13 tables in total)
2. create all fact tables (including temp fact table, 6 tables in total)

1. membershipjoindate_dim_v2

```
CREATE TABLE membershipjoindate_dim_v2
    AS
        SELECT DISTINCT
            joindate AS membershipjoindate
        FROM
            membershipjoinrecords;
```

	MEMBERSHIPJOINDATE
1	23/AUG/07
2	21/MAR/09
3	03/AUG/11
4	10/MAY/10
5	14/OCT/12
6	22/JUN/05
7	29/JUL/06
8	03/DEC/14
9	25/AUG/06
10	16/SEP/05
11	09/JUN/12
12	24/MAY/11
13	18/OCT/12
14	08/OCT/07
15	14/MAY/09

(part of data)

2. membershiptype_dim_v2

```
CREATE TABLE membershiptype_dim_v2
    AS
        SELECT DISTINCT
            membershiptypeid,
            membershipname
        FROM
            membershiptype;
```

	MEMBERSHIPTYPEID	MEMBERSHIPNAME
1	M2	silver
2	M3	Gold
3	M4	Royal
4	M1	bronze

(all data)

3. membershipfeehistory_dim_v2

```
CREATE TABLE membershipfeehistory_dim_v2
    AS
        SELECT
            *
        FROM
            membershipfeehistory;
```

	MEMBERSHIPTYPEID	MEMBERSHIPDISCOUNTFEE	STARTDATE	ENDDATE
1	M4	899.1	01/JAN/05	31/AUG/05
2	M3	719.1	01/JAN/05	31/AUG/05
3	M2	479.2	01/JAN/13	30/APR/13
4	M3	663.17	01/MAR/12	30/APR/12
5	M1	339.15	01/NOV/06	31/MAY/07
6	M4	799.2	01/JAN/13	30/APR/13
7	M3	639.2	01/JAN/13	30/APR/13
8	M1	331.17	01/MAR/08	30/APR/08
9	M4	849.15	01/NOV/06	31/MAY/07
10	M1	331.17	01/MAR/12	30/APR/12
11	M1	359.1	01/JAN/05	31/AUG/05
12	M2	539.1	01/JAN/05	31/AUG/05
13	M2	497.17	01/MAR/08	30/APR/08
14	M2	497.17	01/MAR/12	30/APR/12
15	M3	679.15	01/NOV/06	31/MAY/07
16	M4	829.17	01/MAR/12	30/APR/12
17	M4	829.17	01/MAR/08	30/APR/08
18	M1	319.2	01/JAN/13	30/APR/13
19	M2	509.15	01/NOV/06	31/MAY/07
20	M3	663.17	01/MAR/08	30/APR/08
21	M1	399	01/SEP/05	31/OCT/06
22	M2	599	01/SEP/05	31/OCT/06
23	M3	799	01/SEP/05	31/OCT/06
24	M4	999	01/SEP/05	31/OCT/06
25	M1	399	01/JUN/07	29/FEB/08
26	M2	599	01/JUN/07	29/FEB/08
27	M3	799	01/JUN/07	29/FEB/08
28	M4	999	01/JUN/07	29/FEB/08
29	M1	399	01/MAY/08	29/FEB/12
30	M2	599	01/MAY/08	29/FEB/12
31	M3	799	01/MAY/08	29/FEB/12
32	M4	999	01/MAY/08	29/FEB/12
33	M1	399	01/MAY/12	31/DEC/12
34	M2	599	01/MAY/12	31/DEC/12
35	M3	799	01/MAY/12	31/DEC/12
36	M4	999	01/MAY/12	31/DEC/12
37	M1	399	01/MAY/13	31/DEC/14
38	M2	599	01/MAY/13	31/DEC/14
39	M3	799	01/MAY/13	31/DEC/14
40	M4	999	01/MAY/13	31/DEC/14

(all data)

4. passenger_dim_v2

```
CREATE TABLE passenger_dim_v2
    AS
        SELECT
            passid,
            age as passage,
            firstname,
            lastname,
            nationality as passnationality
        FROM
            passengers;

ALTER TABLE passenger_dim_v2 ADD (
    passengertype        NUMBER,
    passengertypedesc    VARCHAR(20)
);

UPDATE passenger_dim_v2
    SET passengertype = 1, passengertypedesc = 'Children'
WHERE passage < 11;

UPDATE passenger_dim_v2
    SET passengertype = 2, passengertypedesc = 'Teenager'
WHERE passage >= 11 and passage <= 17;

UPDATE passenger_dim_v2
    SET passengertype = 3, passengertypedesc = 'Adult'
WHERE passage >= 18 and passage <= 60;

UPDATE passenger_dim_v2
    SET passengertype = 4, passengertypedesc = 'Elder'
WHERE passage > 60;
```

	PASSID	PASSAGE	FIRSTNAME	LASTNAME	PASSNATIONALITY	PASSENGERTYPE	PASSENGERTYPEDESC
1	1121	40	Londa	Ava	Qatari	3	Adult
2	1122	62	Denny	Keisha	Australian	4	Elder
3	1123	58	Christoper	Dewey	Australian	3	Adult
4	1124	23	Dwain	Zada	Australian	3	Adult
5	1125	13	Cami	Aimee	Australian	2	Teenager
6	1126	43	Sindy	Raymonde	Mauritian	3	Adult
7	1127	79	Wenona	Babara	Liberian	4	Elder
8	1128	2	Murray	Lauralee	Australian	1	Children
9	1129	72	Lien	Opal	Macedonian	4	Elder
10	1130	87	Lavette	Cami	Australian	4	Elder

(part of data)

5. flightclass_dim_v2

```
CREATE TABLE flightclass_dim_v2 (
    flightclass       NUMBER,
    flightclassdesc   VARCHAR(20)
);

INSERT INTO flightclass_dim_v2 VALUES (
    1,
    'First Class'
);

INSERT INTO flightclass_dim_v2 VALUES (
    2,
    'Business Class'
);

INSERT INTO flightclass_dim_v2 VALUES (
    3,
    'Economy Class'
);
```

	FLIGHTCLASS	FLIGHTCLASSDESC
1	1	First Class
2	2	Business Class
3	3	Economy Class

(all data)

6. flight_dim_v2

```
CREATE TABLE flight_dim_v2
    AS
        SELECT
            flightid,
            fare
        FROM
            flights;
```

	FLIGHTID	FARE
1	WN5900	173.72
2	UA5498	656.79
3	SU9794	154.02
4	TP5523	364.84
5	AC7114	305.04
6	MF2590	293.92
7	MU6723	1017.76
8	BR2432	223.68
9	HG3397	410.77
10	FL9200	46.29

(part of data)

7. flightdate_dim_v2

```
CREATE TABLE flightdate_dim_v2
    AS
        SELECT DISTINCT
            flightdate
        FROM
            flights;
```

	FLIGHTDATE
1	15/DEC/09
2	08/SEP/08
3	11/JAN/08
4	11/FEB/08
5	03/FEB/07
6	02/SEP/08
7	24/JUN/08
8	07/JUL/07
9	04/OCT/07
10	04/AUG/08
11	14/JUL/08
12	22/MAR/08
13	08/OCT/07
14	18/OCT/08
15	04/JUN/09

(part of data)

8. flighttype_dim_v2

```
CREATE TABLE flighttype_dim_v2 (
    flighttype       NUMBER,
    flighttypedesc   VARCHAR(20)
);

INSERT INTO flighttype_dim_v2 VALUES (
    1,
    'Domestic'
);

INSERT INTO flighttype_dim_v2 VALUES (
    2,
    'International'
);
```

	FLIGHTTYPE	FLIGHTTYPEDESC
1	1	Domestic
2	2	International

(all data)

9. We build one airport_dim_v2 to include information of Sourceairport_dim_v2 and Destairport_dim_v2

```
CREATE TABLE airport_dim_v2
    AS
        SELECT DISTINCT
            airportid,
            name,
            city,
            country,
            dst
        FROM
            airports;
```

1	2	Madang	Madang	Papua New Guinea	U
2	10	Thule Air Base	Thule	Greenland	E
3	12	Egilsstadir	Egilsstadir	Iceland	N
4	40	Clyde River	Clyde River	Canada	A
5	64	Geraldton Greenstone Regional	Geraldton	Canada	A
6	67	Dryden Rgnl	Dryden	Canada	A
7	88	Mayo	Mayo	Canada	A
8	98	Cold Lake	Cold Lake	Canada	A
9	102	Peace River	Peace River	Canada	A
10	326	Rechlin Larz	Rechlin-laerz	Germany	E
11	330	Jena Schongleina	Jena	Germany	E
12	334	Anklam	Anklam	Germany	E
13	338	Dresden	Dresden	Germany	E
14	345	Dusseldorf	Duesseldorf	Germany	E
15	354	Egelsbach	Egelsbach	Germany	E

(part of data)

10. airlines_dim_v2

```
CREATE TABLE airlines_dim_v2
    AS
        SELECT
            a.airlineid,
            a.name,
            a.alias,
            round(1 / COUNT(p.serviceid),2) AS weight_factor,
            LISTAGG(p.serviceid,
            '_') WITHIN GROUP(
            ORDER BY
                p.serviceid
            ) AS servicegrouplist
        FROM
            airlines a,
            provides p
        WHERE
            a.airlineid = p.airlineid
        GROUP BY
            a.airlineid,
            a.name,
            a.alias;
```

	AIRLINEID	NAME	ALIAS	WEIGHT_FACTOR	SERVICEGROUPLIST
1	1	Private flight	(null)	0.33	1_4_10
2	2	135 Airways	(null)	0.25	1_4_7_8
3	3	1Time Airline	(null)	0.33	3_5_10
4	4	2 Sqn No 1 Elementary Flying Training School	(null)	0.25	3_5_6_9
5	5	213 Flight Unit	(null)	0.25	3_4_6_9
6	6	223 Flight Unit State Airline	(null)	0.25	1_4_7_8
7	7	224th Flight Unit	(null)	0.25	2_5_6_8
8	8	247 Jet Ltd	(null)	0.33	3_4_11
9	9	3D Aviation	(null)	0.33	1_5_10
10	10	40-Mile Air	(null)	0.33	3_5_10
11	11	4D Air	(null)	0.25	2_4_6_9
12	12	611897 Alberta Limited	(null)	0.33	1_4_11
13	13	Ansett Australia	(null)	0.33	3_4_11
14	14	Abacus International	(null)	0.33	3_5_10
15	15	Abelag Aviation	(null)	0.25	3_4_6_9

(part of data)

11. providebridge_dim_v2

```
CREATE TABLE providesbridge_dim_v2
    AS
        SELECT
            *
        FROM
            provides;
```

	AIRLINEID	SERVICEID
1	356	1
2	356	4
3	356	6
4	356	9
5	357	3
6	357	4
7	357	11
8	358	2
9	358	4
10	358	11
11	359	2
12	359	5
13	359	11
14	360	2
15	360	5

(part of data)

12. airlineservices_dim_v2

```
CREATE TABLE airlineservices_dim_v2
    AS
        SELECT
            *
        FROM
            airline_services;
```

	SERVICEID	NAME	DESCRIPTION
1	1	Extra weight 20	Customer can buy up to 20kg extra luggage weight
2	2	Extra weight 10	Customer can buy up to 10kg extra luggage weight
3	3	Extra weight 5	Customer can buy up to 5kg extra luggage weight
4	4	In-flight breakfast	breakfast served to passengers on board
5	5	In-flight meal	meal served to passengers on board
6	6	In-flight fast food	fast food served to passengers on board
7	7	In-flight beverage	drink served to passengers on board
8	8	In-flight movies	Customer can view online movies
9	9	In-flight music	Customer can listen to music
10	10	In-flight games	Customer can play games
11	11	In-flight internet	Customer can access internet via Wifi on board

(all data)

13. flighttime_dim_v2

```
CREATE TABLE flighttime_dim_v2
    AS
        SELECT DISTINCT
            departtime
        FROM
            flights;
```

	FLIGHTTIME
1	02:50:30
2	12:25:24
3	18:30:23
4	03:55:41
5	23:45:04
6	14:25:01
7	05:35:31
8	16:55:46
9	17:50:49
10	10:25:48
11	10:05:35
12	15:00:06
13	19:40:50
14	13:55:18
15	08:15:09

(part of data)

14. Temp_MembershipSales_fact_v2 (for MembershipSales fact table)

```
CREATE TABLE temp_membershipsales_fact_v2
    AS
        SELECT
            m.joindate AS membershipjoindate,
            p.age,
            m.membershiptypeid,
            mh.membershipdiscountfee,
            p.passid
        FROM
            passengers p,
            membershipjoinrecords m,
            membershipfeehistory mh
        WHERE
            p.passid = m.passid
            AND   m.membershiptypeid = mh.membershiptypeid
                AND   m.joindate BETWEEN mh.startdate AND mh.enddate;

--1.2 check if count number of each value of aggregate attribute is only one(no aggregation of level 0)

SELECT
    membershipjoindate,
    passid,
    membershiptypeid,
    count(*),
    SUM(membershipdiscountfee) AS total_membership_sales,
    COUNT(passid) AS total_number_of_membership,
    SUM(age) AS total_membership_age
FROM
    temp_membershipsales_fact_v2
GROUP BY
    membershipjoindate,
    passid,
    membershiptypeid
having
    count(*)>1;
```

	MEMBERSHIPJOINDATE	AGE	MEMBERSHIPTYPEID	MEMBERSHIPDISCOUNTFEE	PASSID
1	25/DEC/13	81	M4	999	5648
2	23/OCT/14	38	M2	599	5658
3	30/MAR/14	23	M3	799	1973
4	05/AUG/06	55	M1	399	5927
5	11/MAR/11	0	M3	799	5014
6	29/MAR/13	6	M4	799.2	6725
7	23/JUL/05	18	M4	899.1	4901
8	23/AUG/07	43	M2	599	4548
9	13/JAN/11	69	M1	399	3456
10	21/MAR/09	6	M4	999	6245
11	22/JAN/14	24	M3	799	1575
12	27/MAR/13	38	M2	479.2	5170
13	28/APR/08	61	M4	829.17	1949
14	24/FEB/13	22	M4	799.2	4918
15	25/JUN/10	83	M2	599	6917

(part of data)

15. MembershipSales_fact_v2 table

```
CREATE TABLE membershipsales_fact_v2
    AS
        SELECT
            membershipjoindate,
            passid,
            membershiptypeid,
            SUM(membershipdiscountfee) AS total_membership_sales,
            COUNT(passid) AS total_number_of_membership,
            SUM(age) AS total_membership_age
        FROM
            temp_membershipsales_fact_v2
        GROUP BY
            membershipjoindate,
            passid,
            membershiptypeid;
```

	MEMBERSHIPJOINDATE	PASSID	MEMBERSHIPTYPEID	TOTAL_MEMBERSHIP_SALES	TOTAL_NUMBER_OF_MEMBERSHIP	TOTAL_MEMBERSHIP_AGE
1	23/JUL/05	4901	M4	899.1	1	18
2	23/AUG/07	4548	M2	599	1	43
3	28/APR/08	1949	M4	829.17	1	61
4	24/FEB/13	4918	M4	799.2	1	22
5	04/OCT/06	5659	M2	599	1	47
6	07/APR/07	5232	M1	339.15	1	34
7	07/FEB/07	3019	M3	679.15	1	28
8	11/APR/05	6344	M2	539.1	1	15
9	04/MAY/10	4724	M2	599	1	0
10	27/APR/06	2573	M2	599	1	82
11	16/NOV/05	4161	M4	999	1	13
12	04/APR/13	1440	M2	479.2	1	3
13	19/NOV/10	2716	M4	999	1	59
14	02/JUL/07	3299	M2	599	1	17
15	12/JAN/07	1972	M3	679.15	1	62

(part of data)

16. Temp_transaction_fact_v2 table (for Transaction fact table)

```
CREATE TABLE temp_transaction_fact_v2
        AS
            SELECT
                t.totalpaid,
                f.fare,
                p.age,
                f.flightdate,
                r.sourceairportid,
                r.destairportid,
                p.passid,
                source_a.country AS source_country,
                dest_a.country AS dest_country,
                f.flightid,
                a.airlineid
            FROM
                routes r,
                airlines a,
                flights f,
                transactions t,
                passengers p,
                airports source_a,
                airports dest_a
            WHERE
                r.routeid = f.routeid
                AND   f.flightid = t.flightid
                    AND   t.passid = p.passid
                        AND   r.sourceairportid = source_a.airportid
                            AND   r.destairportid = dest_a.airportid
                                AND r.airlineid = a.airlineid;
```

```
ALTER TABLE temp_transaction_fact_v2 ADD (
    flighttype      VARCHAR(20),
    flightclass     VARCHAR(20)
);

UPDATE temp_transaction_fact_v2
    SET
        flighttype = 1
WHERE
    source_country = dest_country;

UPDATE temp_transaction_fact_v2
    SET
        flighttype = 2
WHERE
    source_country != dest_country;

UPDATE temp_transaction_fact_v2
    SET
        flightclass = 1
WHERE
    totalpaid >= 2 * fare;

UPDATE temp_transaction_fact_v2
    SET
        flightclass = 2
WHERE
    totalpaid < 2 * fare
    AND   totalpaid >= 1.5 * fare;

UPDATE temp_transaction_fact_v2
    SET
        flightclass = 3
WHERE
    totalpaid < 1.5 * fare;
```

	TOTALPAID	FARE	AGE	FLIGHTDATE	SOURCEAIRPORTID	DESTAIRPORTID	PASSID	SOURCE_COUNTRY	DEST_COUNTRY	FLIGHTID	AIRLINEID	FLIGHTTYPE	FLIGHTCLASS
1	203.48	33.05	14	14/JUL/08	3076	3043	2334	Bangladesh	India	BG2493	13592		1
2	904.47	375.96	8	31/MAY/08	3076	3093	4587	Bangladesh	India	BG2457	13592		1
3	226.54	141.48	85	21/OCT/07	108	156	5241	Canada	Canada	BH5321	26921		2
4	316.62	229.03	35	27/NOV/07	156	108	1814	Canada	Canada	BH5391	26921		3
5	252.78	61.27	34	06/JAN/08	3370	3368	6318	China	China	BK5469	40261		1
6	151.89	79.91	5	12/AUG/08	3393	3368	3699	China	China	BK4687	40261		2
7	187.95	97.07	50	27/APR/08	3371	3379	6884	China	China	BK5964	40261		2
8	467.09	238.7	77	10/MAR/08	3368	3393	6433	China	China	BK4186	40261		2
9	670.22	258.82	6	26/JUL/07	3368	3393	5198	China	China	BK5082	40261		1
10	295.9	89.38	11	19/JUL/08	6187	3205	3471	Vietnam	Vietnam	BL4830	38501		1
11	234.3	157.49	40	02/DEC/08	3393	3382	2669	China	China	8L4618	29421		3
12	482.91	219.47	17	11/NOV/07	3382	3406	6546	China	China	8L5425	29421		1
13	314.47	61.14	79	18/AUG/09	3387	6361	8307	China	China	8L4308	29421		1
14	253.25	221.93	40	25/APR/08	3388	3382	1120	China	China	8L5985	29421		3
15	359.76	76.18	51	05/FEB/08	3373	3382	7044	China	China	8L5331	29421		1

(part of data)

17. Transaction_fact_v2

```
CREATE TABLE transaction_fact_v2
    AS
        SELECT
            flightclass,
            flightdate,
            flighttype,
            sourceairportid,
            destairportid,
            flightid,
            passid,
            airlineid,
            SUM(totalpaid) AS total_ticket_paid,
            COUNT(*) AS total_number_of_ticket,
            SUM(fare) AS total_flight_fare,
            SUM(totalpaid - fare) AS total_agent_profit,
            SUM(age) AS total_passenger_age,
            COUNT(passid) AS total_number_of_passenger
        FROM
            temp_transaction_fact_v2
        GROUP BY
            flightclass,
            flightdate,
            flighttype,
            sourceairportid,
            destairportid,
            flightid,
            passid,
            airlineid;
```

	FLIGHTCLASS	PASSENGERNATIONALITY	PASSENGERTYPE	FLIGHTDATEID	FLIGHTTYPE	SOURCEAIRPORTID	DESTAIRPORTID	AIRLINEID	TOTAL_TICKET_PAID	TOTAL_NUMBER_OF_TICKET	TOTAL_FLIGHT_FARE	TOTAL_AGENT_PROFIT	TOTAL_PASSENGER_AGE	TOTAL_NUMBER_OF_PASSENGER
1	2	Swedish	3	20080427	1	3371	3379	4026	187.95	11	97.07	90.88	50	1
2	3	Australian	3	20081202	1	3393	3382	2942	234.3	11	157.49	76.81	40	1
3	2	Libyan	4	20070316	1	3484	3687	439	247.05	11	157.64	89.41	76	1
4	2	Australian	3	20070923	2	813	879	683	692.85	11	380.89	311.96	39	1
5	1	Australian	3	20080516	2	3320	2276	2091	1830.32	11	808.97	1021.35	40	1
6	3	Australian	1	20080516	2	3320	2276	2091	910.12	11	808.97	101.15	3	1
7	3	Australian	4	20090428	2	2276	3320	2091	4118.74	55	3777.25	341.49	396	5
8	3	Nigerian	3	20090428	2	2276	3320	2091	806.88	11	755.45	51.43	40	1
9	2	Australian	1	20080629	1	3417	6712	55	232.3	11	136.63	95.67	5	1
10	1	Egyptian	4	20090616	1	4144	3379	15999	393.91	11	157.17	236.74	76	1

(part of data)

18. Temp_route_fact_v2 (for Route fact table)

```
CREATE TABLE temp_route_fact_v2
    AS
        SELECT
            r.airlineid,
            f.flightdate,
            r.sourceairportid,
            r.destairportid,
            r.routeid,
            r.distance,
            r.servicecost,
            source_a.country AS source_country,
            dest_a.country AS dest_country,
            f.departtime
        FROM
            flights f,
            routes r,
            airports source_a,
            airports dest_a
        WHERE
            f.routeid = r.routeid
            AND   r.sourceairportid = source_a.airportid
                AND   r.destairportid = dest_a.airportid;
--3.2 create flighttype attribute

ALTER TABLE temp_route_fact_v2 ADD (
    flighttype   VARCHAR(20)
);

UPDATE temp_route_fact_v2
    SET
        flighttype = 1
WHERE
    source_country = dest_country;

UPDATE temp_route_fact_v2
    SET
        flighttype = 2
WHERE
    source_country != dest_country;
```

	AIRLINEID	FLIGHTDATE	SOURCEAIRPORTID	DESTAIRPORTID	ROUTEID	DISTANCE	SERVICECOST	SOURCE_COUNTRY	DEST_COUNTRY	FLIGHTTIME	FLIGHTTYPE
1	2421	14/SEP/08	547	535	12168	472.77	9455.4	United Kingdom	United Kingdom	00:00:31	1
2	2421	24/NOV/08	547	478	12169	249.61	4992.2	United Kingdom	United Kingdom	07:05:34	1
3	2421	17/APR/08	547	478	12169	249.61	4992.2	United Kingdom	United Kingdom	07:15:10	1
4	2421	21/FEB/09	1386	495	12170	363.55	7271	France	United Kingdom	16:30:43	2
5	2421	30/JAN/07	1386	495	12170	363.55	7271	France	United Kingdom	02:30:40	2
6	2421	28/MAY/09	1386	495	12170	363.55	7271	France	United Kingdom	12:10:22	2
7	2421	03/MAR/08	1415	495	12171	321.47	6429.4	France	United Kingdom	10:30:13	2
8	2421	29/AUG/07	1415	495	12171	321.47	6429.4	France	United Kingdom	13:25:07	2
9	2421	10/OCT/08	1415	495	12171	321.47	6429.4	France	United Kingdom	13:00:18	2
10	2421	30/JUN/09	603	534	12172	457.22	9144.4	Ireland	United Kingdom	11:00:56	2
11	2421	18/APR/07	495	1230	12173	1606.78	32135.6	United Kingdom	Spain	06:10:04	2
12	2421	03/JAN/09	495	1212	12174	1409.95	28199	United Kingdom	Spain	02:20:46	2
13	2421	02/AUG/08	495	1212	12174	1409.95	28199	United Kingdom	Spain	06:35:24	2
14	2421	09/OCT/09	495	580	12175	448.47	8969.4	United Kingdom	Netherlands	11:35:41	2
15	2421	07/FEB/09	495	535	12178	571.65	11433	United Kingdom	United Kingdom	07:40:07	1

(part of data)

19. route_fact_v2

```
CREATE TABLE route_fact_v2
    AS
        SELECT
            airlineid,
            flightdate,
            departtime,
            flighttype,
            sourceairportid,
            destairportid,
            COUNT(routeid) AS total_number_of_routes,
            SUM(distance) AS total_route_distance,
            SUM(servicecost) AS total_service_cost
        FROM
            temp_route_fact_v2
        GROUP BY
            airlineid,
            flightdate,
            departtime,
            flighttype,
            sourceairportid,
            destairportid;
```

	AIRLINEID	FLIGHTDATE	FLIGHTTIME	SOURCEAIRPORTID	DESTAIRPORTID	TOTAL_NUMBER_OF_ROUTES	TOTAL_ROUTE_DISTANCE	TOTAL_SERVICE_COST
1	2421	10/NOV/09	04:50:43	2	1386	1	363.55	7271
2	1359	29/JUL/09	04:45:13	1	3076	1	228.85	4577
3	1359	31/MAY/08	02:40:44	2	3093	1	1425.34	28506.8
4	1359	03/JUN/07	07:45:10	2	3316	1	2898.11	57962.2
5	1359	24/JUL/09	06:25:18	2	3069	1	5401.47	108029.4
6	1359	31/MAR/08	00:35:26	2	3076	1	2642.21	52844.2
7	1359	24/FEB/08	15:10:16	2	3076	1	2898.11	57962.2
8	4255	31/DEC/09	12:15:36	2	3304	1	1486.58	29731.6
9	4255	14/FEB/09	01:10:23	2	3406	1	3000.38	60007.6
10	4255	03/MAR/07	20:40:25	2	3272	1	637.62	12752.4
11	4255	04/JUN/07	18:40:01	2	3272	1	1255.17	25103.4
12	4255	16/MAY/09	09:00:53	2	3272	1	3000.38	60007.6
13	4026	09/APR/08	20:35:09	1	3383	1	634.96	12699.2
14	4026	23/MAR/09	18:15:10	1	8979	1	551.59	11031.8
15	4026	09/OCT/08	17:10:20	1	8979	1	551.59	11031.8

(part of data)

Task C.3 Report 1

a. The query questions

What is the top 3 average ages of passengers traveling on business class from an Australian airport?

b. The SQL commands

```
SELECT DISTINCT
    total_passenger_age / total_number_of_passenger AS avg_age
FROM
    transaction_fact_v2 t,
    flightclass_dim_v2 f,
    airport_dim_v2 source_a
WHERE
    t.flightclass = f.flightclass
    AND   t.sourceairportid = source_a.airportid
    AND   f.flightclassdesc = 'Business Class'
    AND   source_a.country = 'Australia'
ORDER BY
    total_passenger_age / total_number_of_passenger DESC
FETCH FIRST 3 ROWS ONLY;
```

c. Screenshots of the query results

AVG_AGE
87
86
85

Task C.3 Report 2

a. The query questions

What is the total number of newly joined gold membership for Adults passenger in each month?

b. The SQL commands

```
SELECT
    TO_CHAR(ms.membershipjoindate,'mm') AS month,
    COUNT(total_number_of_membership) AS total_number_of_memberships
FROM
    passenger_dim_v2 p,
    membershiptype_dim_v2 mt,
    membershipsales_fact_v2 ms
WHERE
    p.passid = ms.passid
    AND   ms.membershiptypeid = mt.membershiptypeid
    AND   p.passengertypedesc = 'Adult'
    AND   mt.membershipname = 'Gold'
GROUP BY
    TO_CHAR(ms.membershipjoindate,'mm')
ORDER BY
    TO_CHAR(ms.membershipjoindate,'mm');
```

c. Screenshots of the query results

```
MO TOTAL_NUMBER_OF_MEMBERSHIPS
-- ----------------------------
01                           140
02                           107
03                           164
04                           109
05                           140
06                           141
07                           146
08                           151
09                           134
10                           152
11                           140

MO TOTAL_NUMBER_OF_MEMBERSHIPS
-- ----------------------------
12                           144

12 rows selected.
```

Task C.4 Report 3: Transactions' report

a. The query questions

Produce the report and maintain all the columns shown in the sample reports.

b. The SQL commands

```
SELECT
    fd.year AS "Flight Year",
    DECODE(GROUPING(ft.flighttypedesc),1,'All Types',ft.flighttypedesc) AS "Flight Type",
    DECODE(GROUPING(fc.flightclassdesc),1,'All Class',fc.flightclassdesc) AS "Flight Class",
    DECODE(GROUPING(source_a.country),1,'All Source Country',source_a.country) AS "Source Country",
    DECODE(GROUPING(dest_a.country),1,'All Destnation Country',dest_a.country) AS "Destination Country",
    SUM(t.total_number_of_ticket) AS "Number of Transactions",
    TO_CHAR(SUM(t.total_agent_profit) / SUM(t.total_number_of_ticket),'9,999,999,999.99') AS "Average Agent Profit(USD)"
FROM
    transaction_fact_v1 t,
    flightclass_dim_v1 fc,
    flightdate_dim_v1 fd,
    flighttype_dim_v1 ft,
    airport_dim_v1 source_a,
    airport_dim_v1 dest_a
WHERE
    t.flightclass = fc.flightclass
    AND   t.flightdateid = fd.flightdateid
    AND   t.flighttype = ft.flighttype
    AND   t.sourceairportid = source_a.airportid
    AND   t.destairportid = dest_a.airportid
GROUP BY
    fd.year,
    CUBE(ft.flighttypedesc,
    fc.flightclassdesc,
    source_a.country,
    dest_a.country);
```

c. Screenshots of the query results

Flig	Flight Type	Flight Class	Source Country	Destination Country	Number of Transactions	Average Agent Pro
2007	All Types	All Class	All Source Country	All Destnation Country	6341	240.55
2007	All Types	All Class	All Source Country	Fiji	34	284.86
2007	All Types	All Class	All Source Country	Guam	1	55.27
2007	All Types	All Class	All Source Country	Iran	1	130.10
2007	All Types	All Class	All Source Country	Iraq	4	57.30
2007	All Types	All Class	All Source Country	Mali	1	537.47
2007	All Types	All Class	All Source Country	Peru	4	122.51
2007	All Types	All Class	All Source Country	Aruba	1	250.78
2007	All Types	All Class	All Source Country	Burma	1	101.80
2007	All Types	All Class	All Source Country	Chile	1	64.22
2007	All Types	All Class	All Source Country	China	166	301.93

Flig	Flight Type	Flight Class	Source Country	Destination Country	Number of Transactions	Average Agent Pro
2007	All Types	All Class	All Source Country	Gabon	1	73.20
2007	All Types	All Class	All Source Country	India	22	170.39
2007	All Types	All Class	All Source Country	Italy	13	302.03
2007	All Types	All Class	All Source Country	Japan	64	351.47
2007	All Types	All Class	All Source Country	Kenya	1	91.61
2007	All Types	All Class	All Source Country	Libya	1	52.49
2007	All Types	All Class	All Source Country	Malta	4	317.82
2007	All Types	All Class	All Source Country	Qatar	1	450.13
2007	All Types	All Class	All Source Country	Samoa	35	254.65
2007	All Types	All Class	All Source Country	Spain	20	224.50
2007	All Types	All Class	All Source Country	Yemen	1	57.94

This is only part of query result. The result has 10790 rows in total.

Task C.4 Report 4: Report with proper sub-totals

a. The query questions

What are the sub-total and total agent profits of airports and airlines? (You must use the Cube operator)

b. The SQL commands

```
SELECT
    DECODE(GROUPING(a.name),1,'All Airlines',a.name) AS airline_name,
    DECODE(GROUPING(source_a.name),1,'All Source Airports',source_a.name) AS source_airport_name,
    TO_CHAR(SUM(total_agent_profit),'9,999,999,999.99') AS total_agent_profit
FROM
    transaction_fact_v1 t,
    airport_dim_v1 source_a,
    airlines_dim_v1 a
WHERE
    source_a.airportid = t.sourceairportid
    AND   a.airlineid = t.airlineid
GROUP BY
    CUBE(a.name,
    source_a.name);
```

c. Screenshots of the query results

AIRLINE_NAME	SOURCE_AIRPORT_NAME	TOTAL_AGENT_PROFI
All Airlines	All Source Airports	6,103,628.71
All Airlines	Goa	103.99
All Airlines	Osh	491.50
All Airlines	Ufa	54.85
All Airlines	Bari	271.75
All Airlines	Bodo	270.27
All Airlines	City	841.68
All Airlines	Cork	572.20
All Airlines	Faro	86.60
All Airlines	Gaya	66.51
All Airlines	Hato	1,130.15

AIRLINE_NAME	SOURCE_AIRPORT_NAME	TOTAL_AGENT_PROFI
All Airlines	Juba	261.62
All Airlines	Kone	254.57
All Airlines	Luqa	173.90
All Airlines	Naha	741.27
All Airlines	Novy	86.37
All Airlines	Orly	1,369.50
All Airlines	Paro	372.67
All Airlines	Pisa	409.67
All Airlines	Pune	1,020.61
All Airlines	Sale	27.82
All Airlines	Sola	740.53

This is only part of query result. The result has 3378 rows in total.

Task C.4 Report 5: Report with moving and cumulative aggregates

a. The query questions

What are the total and cumulative monthly total sales of Gold membership in 2009?

b. The SQL commands

```
SELECT
    md.membershipjoinmonth AS month,
    TO_CHAR(SUM(total_membership_sales),'9,999,999,999.99') AS total_sales,
    TO_CHAR(SUM(SUM(total_membership_sales) ) OVER(
        ORDER BY
            md.membershipjoinmonth
        ROWS UNBOUNDED PRECEDING
    ),'9,999,999,999.99') AS cum_sales,
    TO_CHAR(AVG(SUM(total_membership_sales) ) OVER(
        ORDER BY
            md.membershipjoinmonth
        ROWS 2 PRECEDING
    ),'9,999,999,999.99') AS moving_2_month_avg
FROM
    membershipsales_fact_v1 ms,
    membershiptype_dim_v1 mt,
    membershipjoindate_dim_v1 md
WHERE
    ms.membershiptypeid = mt.membershiptypeid
    AND   ms.membershipjoindateid = md.membershipjoindateid
    AND   md.membershipjoinyear = '2009'
    AND   mt.membershipname = 'Gold'
GROUP BY
    md.membershipjoinmonth;
```

c. Screenshots of the query results

```
MO TOTAL_SALES       CUM_SALES         MOVING_2_MONTH_AV
-- ----------------- ----------------- -----------------
01         24,769.00         24,769.00         24,769.00
02         16,779.00         41,548.00         20,774.00
03         23,970.00         65,518.00         21,839.33
04         24,769.00         90,287.00         21,839.33
05         23,970.00        114,257.00         24,236.33
06         23,171.00        137,428.00         23,970.00
07         23,970.00        161,398.00         23,703.67
08         26,367.00        187,765.00         24,502.67
09         13,583.00        201,348.00         21,306.67
10         21,573.00        222,921.00         20,507.67
11         14,382.00        237,303.00         16,512.67

MO TOTAL_SALES       CUM_SALES         MOVING_2_MONTH_AV
-- ----------------- ----------------- -----------------
12         28,764.00        266,067.00         21,573.00

12 rows selected.
```

Task C.4 Report 6: Report with partition

a. The query questions

What are the ranks by total number of incoming routes in each country?

b. The SQL commands

```
SELECT
    dest_a.country AS country,
    dest_a.city AS city,
    SUM(total_number_of_routes) AS total_incoming_routes,
    RANK() OVER(
        PARTITION BY dest_a.country
        ORDER BY
            SUM(total_number_of_routes) DESC
    ) AS city_rank
FROM
    airport_dim_v1 dest_a,
    route_fact_v1 r
WHERE
    dest_a.airportid = r.destairportid
GROUP BY
    dest_a.country,
    dest_a.city;
```

c. Screenshots of the query results

COUNTRY	CITY	TOTAL_INCOMING_ROUTES	CITY_RANK
Afghanistan	Kabul	23	1
Afghanistan	Herat	6	2
Afghanistan	Kandahar	4	3
Albania	Tirana	23	1
Algeria	Algier	57	1
Algeria	Oran	14	2
Algeria	Setif	11	3
Algeria	Annaba	10	4
Algeria	Constantine	8	5
Algeria	Tlemcen	5	6
Algeria	Batna	5	6

COUNTRY	CITY	TOTAL_INCOMING_ROUTES	CITY_RANK
Algeria	Bejaja	5	6
Algeria	Biskra	3	9
Algeria	Illizi	3	9
Algeria	Djanet	3	9
Algeria	Ghardaia	3	9
Algeria	Tamanrasset	3	9
Algeria	Ouargla	1	14
American Samoa	Pago Pago	2	1
Angola	Luanda	37	1
Angola	Lubango	6	2
Angola	Ondjiva	4	3

This is only part of query result. The result has 2613 rows in total.

Task C.5 (for each Report)

Report 3

a. SQL for original query of report 3

```
EXPLAIN PLAN FOR
SELECT
    fd.year AS "Flight Year",
    DECODE(GROUPING(ft.flighttypedesc),1,'All Types',ft.flighttypedesc) AS "Flight Type",
    DECODE(GROUPING(fc.flightclassdesc),1,'All Class',fc.flightclassdesc) AS "Flight Class",
    DECODE(GROUPING(source_a.country),1,'All Source Country',source_a.country) AS "Source Country",
    DECODE(GROUPING(dest_a.country),1,'All Destination Country',dest_a.country) AS "Destination Country",
    SUM(t.total_number_of_ticket) AS "Number of Transactions",
    TO_CHAR(SUM(t.total_agent_profit) / SUM(t.total_number_of_ticket),'9,999,999,999.99') AS "Average Agent Profit(USD)"
FROM
    transaction_fact_v1 t,
    flightclass_dim_v1 fc,
    flightdate_dim_v1 fd,
    flighttype_dim_v1 ft,
    airport_dim_v1 source_a,
    airport_dim_v1 dest_a
WHERE
    t.flightclass = fc.flightclass
    AND   t.flightdateid = fd.flightdateid
    AND   t.flighttype = ft.flighttype
    AND   t.sourceairportid = source_a.airportid
    AND   t.destairportid = dest_a.airportid
GROUP BY
    fd.year,
    CUBE(ft.flighttypedesc,
    fc.flightclassdesc,
    source_a.country,
    dest_a.country);

SELECT * FROM TABLE ( dbms_xplan.display );
```

b. Screenshot of the query result

Flig	Flight Type	Flight Class	Source Country	Destination Country	Number of Transactions	Average Agent Pro
2007	All Types	All Class	All Source Country	All Destnation Country	6341	240.55
2007	All Types	All Class	All Source Country	Fiji	34	284.86
2007	All Types	All Class	All Source Country	Guam	1	55.27
2007	All Types	All Class	All Source Country	Iran	1	130.10
2007	All Types	All Class	All Source Country	Iraq	4	57.30
2007	All Types	All Class	All Source Country	Mali	1	537.47
2007	All Types	All Class	All Source Country	Peru	4	122.51
2007	All Types	All Class	All Source Country	Aruba	1	250.78
2007	All Types	All Class	All Source Country	Burma	1	101.80
2007	All Types	All Class	All Source Country	Chile	1	64.22
2007	All Types	All Class	All Source Country	China	166	301.93

Flig	Flight Type	Flight Class	Source Country	Destination Country	Number of Transactions	Average Agent Pro
2007	All Types	All Class	All Source Country	Gabon	1	73.20
2007	All Types	All Class	All Source Country	India	22	170.39
2007	All Types	All Class	All Source Country	Italy	13	302.03
2007	All Types	All Class	All Source Country	Japan	64	351.47
2007	All Types	All Class	All Source Country	Kenya	1	91.61
2007	All Types	All Class	All Source Country	Libya	1	52.49
2007	All Types	All Class	All Source Country	Malta	4	317.82
2007	All Types	All Class	All Source Country	Qatar	1	450.13
2007	All Types	All Class	All Source Country	Samoa	35	254.65
2007	All Types	All Class	All Source Country	Spain	20	224.50
2007	All Types	All Class	All Source Country	Yemen	1	57.94

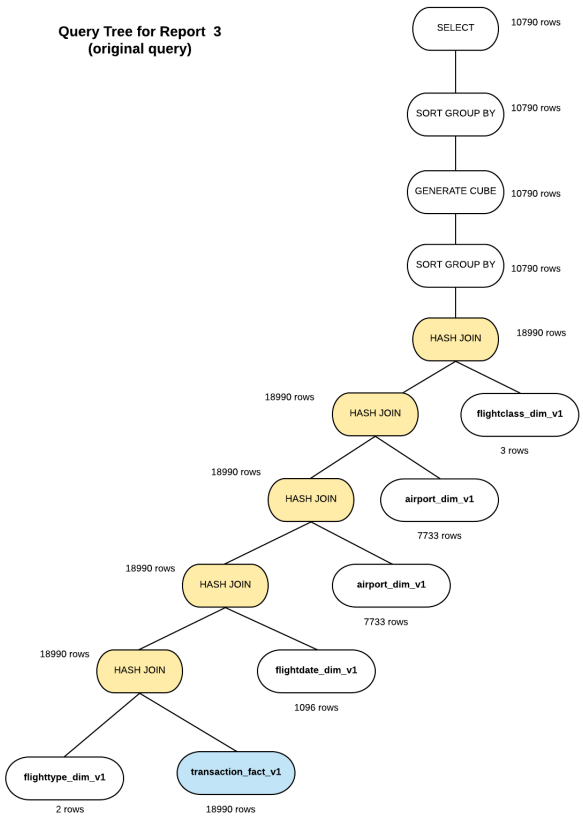
This is only part of query result. The result has 10790 rows in total.

c. Execution plan of the original query

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 1337226414

-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                     | 10790 |  1085K|       |   420   (1)| 00:00:01 |
|   1 |  SORT GROUP BY           |                     | 10790 |  1085K|       |   420   (1)| 00:00:01 |
|   2 |   GENERATE CUBE          |                     | 10790 |  1085K|       |   420   (1)| 00:00:01 |
|   3 |    SORT GROUP BY         |                     | 10790 |  1085K|  2120K|   420   (1)| 00:00:01 |
|*  4 |     HASH JOIN            |                     | 18990 |  1910K|       |    97   (2)| 00:00:01 |
|   5 |      TABLE ACCESS FULL   | FLIGHTCLASS_DIM_V1  |     3 |    51 |       |     3   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
|*  6 |      HASH JOIN           |                     | 18990 |  1594K|       |    94   (2)| 00:00:01 |
|   7 |       TABLE ACCESS FULL  | AIRPORT_DIM_V1      |  7733 |   105K|       |    17   (0)| 00:00:01 |
|*  8 |       HASH JOIN          |                     | 18990 |  1335K|       |    77   (2)| 00:00:01 |
|   9 |        TABLE ACCESS FULL | AIRPORT_DIM_V1      |  7733 |   105K|       |    17   (0)| 00:00:01 |
|* 10 |        HASH JOIN         |                     | 18990 |  1075K|       |    59   (0)| 00:00:01 |
|  11 |         TABLE ACCESS FULL| FLIGHTDATE_DIM_V1   |  1096 | 15344 |       |     4   (0)| 00:00:01 |
|* 12 |         HASH JOIN        |                     | 18990 |   815K|       |    55   (0)| 00:00:01 |
|  13 |          TABLE ACCESS FULL| FLIGHTTYPE_DIM_V1  |     2 |    30 |       |     3   (0)| 00:00:01 |
|  14 |          TABLE ACCESS FULL| TRANSACTION_FACT_V1| 18990 |   537K|       |    52   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("FC"."FLIGHTCLASS"=TO_NUMBER("T"."FLIGHTCLASS"))
   6 - access("T"."DESTAIRPORTID"="DEST_A"."AIRPORTID")
   8 - access("T"."SOURCEAIRPORTID"="SOURCE_A"."AIRPORTID")
  10 - access("T"."FLIGHTDATEID"="FD"."FLIGHTDATEID")
  12 - access("FT"."FLIGHTTYPE"=TO_NUMBER("T"."FLIGHTTYPE"))

Note
-----

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
   - dynamic statistics used: dynamic sampling (level=2)
   - 1 Sql Plan Directive used for this statement

35 rows selected.
```

From the execution plan, we know there are 15 operations and the total rows being processed are quite large. Then we plot query tree to observe this execution plan.

d. Query Tree of the original query



From the query tree, we find the system choose to use HASH JOIN for all tables because these tables don't have index. But in these fact and dimension tables, we find there are two table only having few rows. We can change our query to add some hints to optimize this query.

e. New Query

We notice that FLIGHTTYPE dimension and FLIGHTCLASS dimension only have 2 and 3 rows. So, we can use Cartesian product to merge them first and then HASH JOIN other tables.

```
EXPLAIN PLAN FOR
SELECT /*+ ORDERED */
    fd.year AS "Flight Year",
    DECODE(GROUPING(ft.flighttypedesc),1,'All Types',ft.flighttypedesc) AS "Flight Type",
    DECODE(GROUPING(fc.flightclassdesc),1,'All Class',fc.flightclassdesc) AS "Flight Class",
    DECODE(GROUPING(source_a.country),1,'All Source Country',source_a.country) AS "Source Country",
    DECODE(GROUPING(dest_a.country),1,'All Destination Country',dest_a.country) AS "Destination Country",
    SUM(t.total_number_of_ticket) AS "Number of Transactions",
    TO_CHAR(SUM(t.total_agent_profit) / SUM(t.total_number_of_ticket),'9,999,999,999.99') AS "Average Agent Profit(USD)"
FROM
    flighttype_dim_v1 ft,
    flightclass_dim_v1 fc,
    transaction_fact_v1 t,
    flightdate_dim_v1 fd,
    airport_dim_v1 source_a,
    airport_dim_v1 dest_a
WHERE
    t.flightclass = fc.flightclass
    AND   t.flightdateid = fd.flightdateid
    AND   t.flighttype = ft.flighttype
    AND   t.sourceairportid = source_a.airportid
    AND   t.destairportid = dest_a.airportid
GROUP BY
    fd.year,
    CUBE(ft.flighttypedesc,
    fc.flightclassdesc,
    source_a.country,
    dest_a.country);

SELECT * FROM TABLE ( dbms_xplan.display );
```

f. Result of new query

Flig	Flight Type	Flight Class	Source Country	Destination Country	Number of Transactions	Average Agent Pro
2007	All Types	All Class	All Source Country	All Destnation Country	6341	240.55
2007	All Types	All Class	All Source Country	Fiji	34	284.86
2007	All Types	All Class	All Source Country	Guam	1	55.27
2007	All Types	All Class	All Source Country	Iran	1	130.10
2007	All Types	All Class	All Source Country	Iraq	4	57.30
2007	All Types	All Class	All Source Country	Mali	1	537.47
2007	All Types	All Class	All Source Country	Peru	4	122.51
2007	All Types	All Class	All Source Country	Aruba	1	250.78
2007	All Types	All Class	All Source Country	Burma	1	101.80
2007	All Types	All Class	All Source Country	Chile	1	64.22
2007	All Types	All Class	All Source Country	China	166	301.93

Flig	Flight Type	Flight Class	Source Country	Destination Country	Number of Transactions	Average Agent Pro
2007	All Types	All Class	All Source Country	Gabon	1	73.20
2007	All Types	All Class	All Source Country	India	22	170.39
2007	All Types	All Class	All Source Country	Italy	13	302.03
2007	All Types	All Class	All Source Country	Japan	64	351.47
2007	All Types	All Class	All Source Country	Kenya	1	91.61
2007	All Types	All Class	All Source Country	Libya	1	52.49
2007	All Types	All Class	All Source Country	Malta	4	317.82
2007	All Types	All Class	All Source Country	Qatar	1	450.13
2007	All Types	All Class	All Source Country	Samoa	35	254.65
2007	All Types	All Class	All Source Country	Spain	20	224.50
2007	All Types	All Class	All Source Country	Yemen	1	57.94

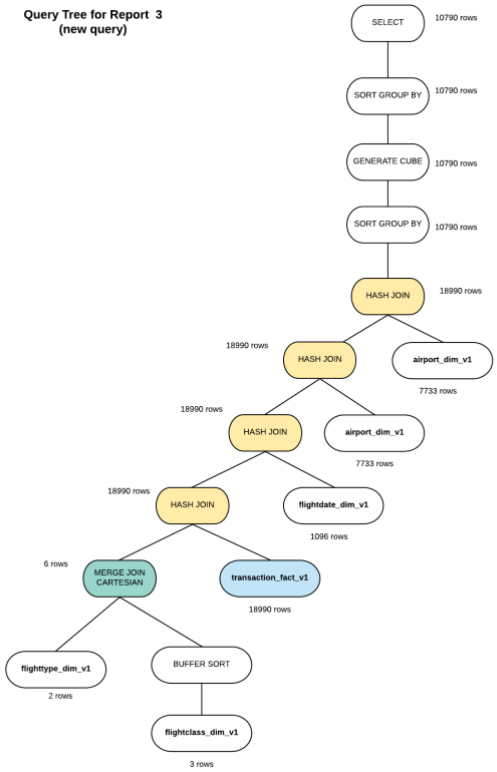
We can see the result of new query is the same as original one. And the total number of rows selected is also 10790.

g. Execution Plan of new query

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 575675007

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name               | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                    | 10790 |  1085K|       |   421   (1)| 00:00:01 |
|   1 |  SORT GROUP BY             |                    | 10790 |  1085K|       |   421   (1)| 00:00:01 |
|   2 |   GENERATE CUBE            |                    | 10790 |  1085K|       |   421   (1)| 00:00:01 |
|   3 |    SORT GROUP BY           |                    | 10790 |  1085K|  2120K|   421   (1)| 00:00:01 |
|*  4 |     HASH JOIN              |                    | 18990 |  1910K|       |    98   (2)| 00:00:01 |
|   5 |      TABLE ACCESS FULL     | AIRPORT_DIM_V1     |  7733 |   105K|       |    17   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
|*  6 |       HASH JOIN            |                    | 18990 |  1650K|       |    81   (2)| 00:00:01 |
|   7 |        TABLE ACCESS FULL   | AIRPORT_DIM_V1     |  7733 |   105K|       |    17   (0)| 00:00:01 |
|*  8 |        HASH JOIN           |                    | 18990 |  1390K|       |    63   (0)| 00:00:01 |
|   9 |         TABLE ACCESS FULL  | FLIGHTDATE_DIM_V1  |  1096 | 15344 |       |     4   (0)| 00:00:01 |
|* 10 |         HASH JOIN          |                    | 18990 |  1131K|       |    59   (0)| 00:00:01 |
|  11 |          MERGE JOIN CARTESIAN|                  |     6 |   192 |       |     7   (0)| 00:00:01 |
|  12 |           TABLE ACCESS FULL| FLIGHTTYPE_DIM_V1  |     2 |    30 |       |     3   (0)| 00:00:01 |
|  13 |           BUFFER SORT      |                    |     3 |    51 |       |     4   (0)| 00:00:01 |
|  14 |            TABLE ACCESS FULL| FLIGHTCLASS_DIM_V1|     3 |    51 |       |     2   (0)| 00:00:01 |
|  15 |          TABLE ACCESS FULL | TRANSACTION_FACT_V1| 18990 |   537K|       |    52   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("T"."DESTAIRPORTID"="DEST_A"."AIRPORTID")
   6 - access("T"."SOURCEAIRPORTID"="SOURCE_A"."AIRPORTID")
   8 - access("T"."FLIGHTDATEID"="FD"."FLIGHTDATEID")
  10 - access("FC"."FLIGHTCLASS"=TO_NUMBER("T"."FLIGHTCLASS") AND
              "FT"."FLIGHTTYPE"=TO_NUMBER("T"."FLIGHTTYPE"))

Note

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - 1 Sql Plan Directive used for this statement

36 rows selected.
```

From the execution plan, we know there are 16 operations and number of records being processed has been reduced.

h. Query Tree of the new query



From the query tree, we find there is a Cartesian Product between flighttype and flightclass dimensions. That have reduced the number of records being processed.

i. Compare new query and original one

When we compare to two queries, we focus on their number of records being processed. The small one is better because the processing time is faster. Compared to original query, we find that number of records being processed in execution plan of new query is less. So, we think **the new query is more efficient than original query**.

Report 4

a. SQL for report 4

```
EXPLAIN PLAN FOR
SELECT
    DECODE(GROUPING(a.name),1,'All Airlines',a.name) AS airline_name,
    DECODE(GROUPING(source_a.name),1,'All Source Airports',source_a.name) AS source_airport_name,
    TO_CHAR(SUM(total_agent_profit),'9,999,999,999.99') AS total_agent_profit
FROM
    transaction_fact_v1 t,
    airport_dim_v1 source_a,
    airlines_dim_v1 a
WHERE
    source_a.airportid = t.sourceairportid
    AND   a.airlineid = t.airlineid
GROUP BY
    CUBE(a.name,
    source_a.name);

SELECT * FROM TABLE ( dbms_xplan.display );
```

b. Screenshot of the query result

AIRLINE_NAME	SOURCE_AIRPORT_NAME	TOTAL_AGENT_PROFI
All Airlines	All Source Airports	6,103,628.71
All Airlines	Goa	103.99
All Airlines	Osh	491.50
All Airlines	Ufa	54.85
All Airlines	Bari	271.75
All Airlines	Bodo	270.27
All Airlines	City	841.68
All Airlines	Cork	572.20
All Airlines	Faro	86.60
All Airlines	Gaya	66.51
All Airlines	Hato	1,130.15

AIRLINE_NAME	SOURCE_AIRPORT_NAME	TOTAL_AGENT_PROFI
All Airlines	Juba	261.62
All Airlines	Kone	254.57
All Airlines	Luqa	173.90
All Airlines	Naha	741.27
All Airlines	Novy	86.37
All Airlines	Orly	1,369.50
All Airlines	Paro	372.67
All Airlines	Pisa	409.67
All Airlines	Pune	1,020.61
All Airlines	Sale	27.82
All Airlines	Sola	740.53

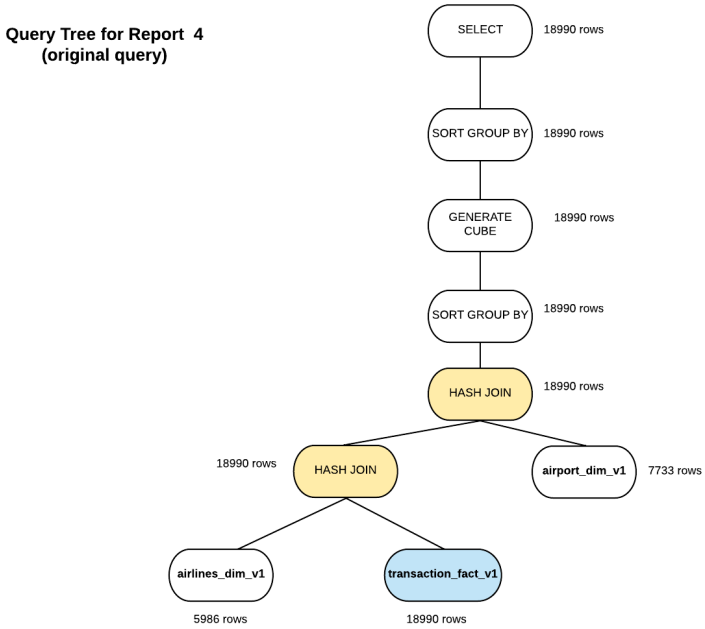
This is only part of query result. The result has 3378 rows in total.

c. Execution plan of the original query

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------
Plan hash value: 2543844891

-------------------------------------------------------------------------------------------------------------
| Id  | Operation             | Name               | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                    | 18990 |  1057K|       |   347   (1)| 00:00:01 |
|   1 |  SORT GROUP BY        |                    | 18990 |  1057K|       |   347   (1)| 00:00:01 |
|   2 |   GENERATE CUBE       |                    | 18990 |  1057K|       |   347   (1)| 00:00:01 |
|   3 |    SORT GROUP BY      |                    | 18990 |  1057K|  1280K|   347   (1)| 00:00:01 |
|*  4 |     HASH JOIN         |                    | 18990 |  1057K|       |    82   (2)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| AIRPORT_DIM_V1     |  7733 |   151K|       |    17   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------
|*  6 |      HASH JOIN        |                    | 18990 |   686K|       |    64   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| AIRLINES_DIM_V1    |  5986 |   140K|       |    12   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| TRANSACTION_FACT_V1| 18990 |   241K|       |    52   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("SOURCE_A"."AIRPORTID"="T"."SOURCEAIRPORTID")
   6 - access("A"."AIRLINEID"="T"."AIRLINEID")

21 rows selected.
```

From the execution plan, we know there are 9 operations. Then we plot query tree to observe this execution plan.

d. Query Tree of the original query



From the query tree of original query, we find system use HASH JOIN method to join tables because there is no index in all tables.

e. New Query

We find execution plan of original query is pretty good. And we want to change join into Cartesian product method to find the difference between original query and new ones.

```
EXPLAIN PLAN FOR
SELECT /*+ ORDERED */
    DECODE(GROUPING(a.name),1,'All Airlines',a.name) AS airline_name,
    DECODE(GROUPING(source_a.name),1,'All Source Airports',source_a.name) AS source_airport_name,
    TO_CHAR(SUM(total_agent_profit),'9,999,999,999.99') AS total_agent_profit
FROM
    airport_dim_v1 source_a,
    airlines_dim_v1 a,
    transaction_fact_v1 t
WHERE
    source_a.airportid = t.sourceairportid
    AND   a.airlineid = t.airlineid
GROUP BY
    CUBE(a.name,
    source_a.name);

SELECT * FROM TABLE ( dbms_xplan.display );
```

f. Result of new query

AIRLINE_NAME	SOURCE_AIRPORT_NAME	TOTAL_AGENT_PROFI
All Airlines	All Source Airports	6,103,628.71
All Airlines	Goa	103.99
All Airlines	Osh	491.50
All Airlines	Ufa	54.85
All Airlines	Bari	271.75
All Airlines	Bodo	270.27
All Airlines	City	841.68
All Airlines	Cork	572.20
All Airlines	Faro	86.60
All Airlines	Gaya	66.51
All Airlines	Hato	1,130.15

AIRLINE_NAME	SOURCE_AIRPORT_NAME	TOTAL_AGENT_PROFI
All Airlines	Juba	261.62
All Airlines	Kone	254.57
All Airlines	Luga	173.90
All Airlines	Naha	741.27
All Airlines	Novy	86.37
All Airlines	Orly	1,369.50
All Airlines	Paro	372.67
All Airlines	Pisa	409.67
All Airlines	Pune	1,020.61
All Airlines	Sale	27.82
All Airlines	Sola	740.53

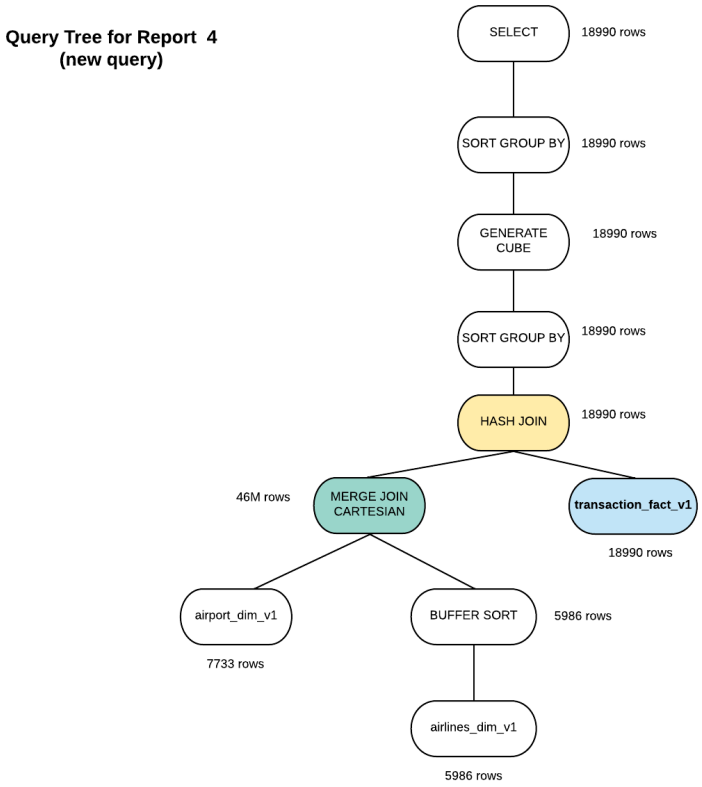
We find the result of new query is same as original ones. The result has 3378 rows in total.

g. Execution Plan of new query

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------
Plan hash value: 986304251

-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                     | 18990 |  1057K|       | 80332   (1)| 00:00:04 |
|   1 |  SORT GROUP BY           |                     | 18990 |  1057K|       | 80332   (1)| 00:00:04 |
|   2 |   GENERATE CUBE          |                     | 18990 |  1057K|       | 80332   (1)| 00:00:04 |
|   3 |    SORT GROUP BY         |                     | 18990 |  1057K|  1280K| 80332   (1)| 00:00:04 |
|*  4 |     HASH JOIN            |                     | 18990 |  1057K|       | 80066   (1)| 00:00:04 |
|   5 |      TABLE ACCESS FULL   | TRANSACTION_FACT_V1 | 18990 |   241K|       |    52   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------
|   6 |      MERGE JOIN CARTESIAN|                     |    46M|  1942M|       | 79884   (1)| 00:00:04 |
|   7 |       TABLE ACCESS FULL  | AIRPORT_DIM_V1      |  7733 |   151K|       |    17   (0)| 00:00:01 |
|   8 |       BUFFER SORT        |                     |  5986 |   140K|       | 79867   (1)| 00:00:04 |
|   9 |        TABLE ACCESS FULL | AIRLINES_DIM_V1     |  5986 |   140K|       |    10   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("SOURCE_A"."AIRPORTID"="T"."SOURCEAIRPORTID" AND "A"."AIRLINEID"="T"."AIRLINEID")

21 rows selected.
```

From the execution plan, we know there are 10 operations and the total number of rows being processed is larger than before. Then we plot query tree to observe this execution plan.

h. Query Tree of the new query



From the query tree of new query, we find system change HASH JOIN into Cartesian product methods and then make number of records being processed quite large.

i. Compare two query and find better one

Compared to original query, we find execution plan of new query has more operations. And at the same time, number of records become much larger than original query because of Cartesian Product method. Therefore, we think **original query is more efficient than new query**.

Report 5

a. SQL for report 5

```
EXPLAIN PLAN FOR
SELECT
    md.membershipjoinmonth AS month,
    TO_CHAR(SUM(total_membership_sales),'9,999,999,999.99') AS total_sales,
    TO_CHAR(SUM(SUM(total_membership_sales) ) OVER(
        ORDER BY
            md.membershipjoinmonth
        ROWS UNBOUNDED PRECEDING
    ),'9,999,999,999.99') AS cum_sales,
    TO_CHAR(AVG(SUM(total_membership_sales) ) OVER(
        ORDER BY
            md.membershipjoinmonth
        ROWS 2 PRECEDING
    ),'9,999,999,999.99') AS moving_2_month_avg
FROM
    membershipsales_fact_v1 ms,
    membershiptype_dim_v1 mt,
    membershipjoindate_dim_v1 md
WHERE
    ms.membershiptypeid = mt.membershiptypeid
    AND   ms.membershipjoindateid = md.membershipjoindateid
    AND   md.membershipjoinyear = '2009'
    AND   mt.membershipname = 'Gold'
GROUP BY
    md.membershipjoinmonth;

SELECT * FROM TABLE ( dbms_xplan.display );
```

b. Screenshot of the query result

```
MO TOTAL_SALES        CUM_SALES          MOVING_2_MONTH_AV
-- ------------------ ------------------ ------------------
01          24,769.00          24,769.00          24,769.00
02          16,779.00          41,548.00          20,774.00
03          23,970.00          65,518.00          21,839.33
04          24,769.00          90,287.00          21,839.33
05          23,970.00         114,257.00          24,236.33
06          23,171.00         137,428.00          23,970.00
07          23,970.00         161,398.00          23,703.67
08          26,367.00         187,765.00          24,502.67
09          13,583.00         201,348.00          21,306.67
10          21,573.00         222,921.00          20,507.67
11          14,382.00         237,303.00          16,512.67

MO TOTAL_SALES        CUM_SALES          MOVING_2_MONTH_AV
-- ------------------ ------------------ ------------------
12          28,764.00         266,067.00          21,573.00

12 rows selected.
```

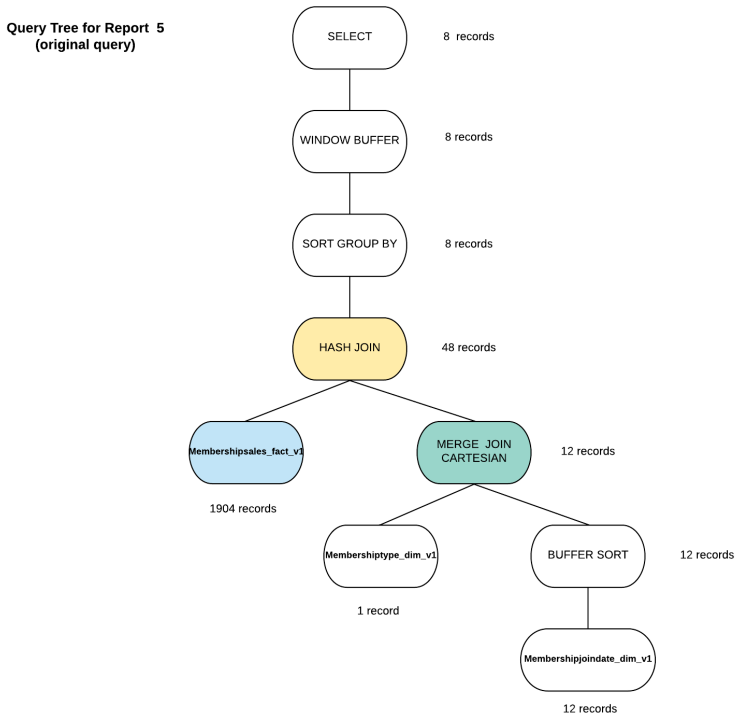
This is the query result. The result has 12 rows in total.

c. Execution plan of the original query

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 28405488

-------------------------------------------------------------------------------------------------------------------------
| Id  | Operation               | Name                    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                         |     8 |   312 |    12   (9)| 00:00:01 |
|   1 |  WINDOW BUFFER          |                         |     8 |   312 |    12   (9)| 00:00:01 |
|   2 |   SORT GROUP BY         |                         |     8 |   312 |    12   (9)| 00:00:01 |
|*  3 |    HASH JOIN            |                         |    48 |  1872 |    11   (0)| 00:00:01 |
|   4 |     MERGE JOIN CARTESIAN|                         |    12 |   288 |     6   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL  | MEMBERSHIPTYPE_DIM_V1   |     1 |     9 |     3   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
|   6 |      BUFFER SORT        |                         |    12 |   180 |     3   (0)| 00:00:01 |
|*  7 |       TABLE ACCESS FULL | MEMBERSHIPJOINDATE_DIM_V1 |    12 |   180 |     3   (0)| 00:00:01 |
|   8 |     TABLE ACCESS FULL   | MEMBERSHIPSALES_FACT_V1 |  1904 | 28560 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("MS"."MEMBERSHIPTYPEID"="MT"."MEMBERSHIPTYPEID" AND
              "MS"."MEMBERSHIPJOINDATEID"="MD"."MEMBERSHIPJOINDATEID")
   5 - filter("MT"."MEMBERSHIPNAME"='Gold')

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
   7 - filter("MD"."MEMBERSHIPJOINYEAR"='2009')

23 rows selected.
```

From the execution plan, we know there are 9 operations. Then we plot query tree to observe this execution plan.

d. Query Tree of the original query



From the query tree of original query, we find “WINDOW BUFFER” and “SORT GROUP BY” are inside because of cumulative and moving aggregate queries.

e. New Query

We find there is “SORT GROUP BY” operation in original query. But we know this operation is worse than first group by and then sort method. Then we want to use a splitting query technique to solve it.

```
EXPLAIN PLAN FOR
SELECT /*+ no_merge */
    innerquery.month,
    innerquery.total_sales,
    TO_CHAR(SUM(innerquery.total_sales) OVER(
        ORDER BY
            innerquery.month
        ROWS UNBOUNDED PRECEDING
    ),'9,999,999,999.99') AS cum_sales,
    TO_CHAR(AVG(innerquery.total_sales) OVER(
        ORDER BY
            innerquery.month
        ROWS 2 PRECEDING
    ),'9,999,999,999.99') AS moving_2_month_avg
FROM
    ( (
        SELECT
            md.membershipjoinmonth AS month,
            SUM(total_membership_sales) AS total_sales
        FROM
            membershipsales_fact_v1 ms,
            membershiptype_dim_v1 mt,
            membershipjoindate_dim_v1 md
        WHERE
            ms.membershiptypeid = mt.membershiptypeid
            AND   ms.membershipjoindateid = md.membershipjoindateid
            AND   md.membershipjoinyear = '2009'
            AND   mt.membershipname = 'Gold'
        GROUP BY
            md.membershipjoinmonth
    ) innerquery );

SELECT * FROM TABLE ( dbms_xplan.display );
```

f. Result of new query

```
MO TOTAL_SALES CUM_SALES         MOVING_2_MONTH_AV
-- ----------- ----------------- -----------------
01       24769         24,769.00         24,769.00
02       16779         41,548.00         20,774.00
03       23970         65,518.00         21,839.33
04       24769         90,287.00         21,839.33
05       23970        114,257.00         24,236.33
06       23171        137,428.00         23,970.00
07       23970        161,398.00         23,703.67
08       26367        187,765.00         24,502.67
09       13583        201,348.00         21,306.67
10       21573        222,921.00         20,507.67
11       14382        237,303.00         16,512.67

MO TOTAL_SALES CUM_SALES         MOVING_2_MONTH_AV
-- ----------- ----------------- -----------------
12       28764        266,067.00         21,573.00

12 rows selected.
```

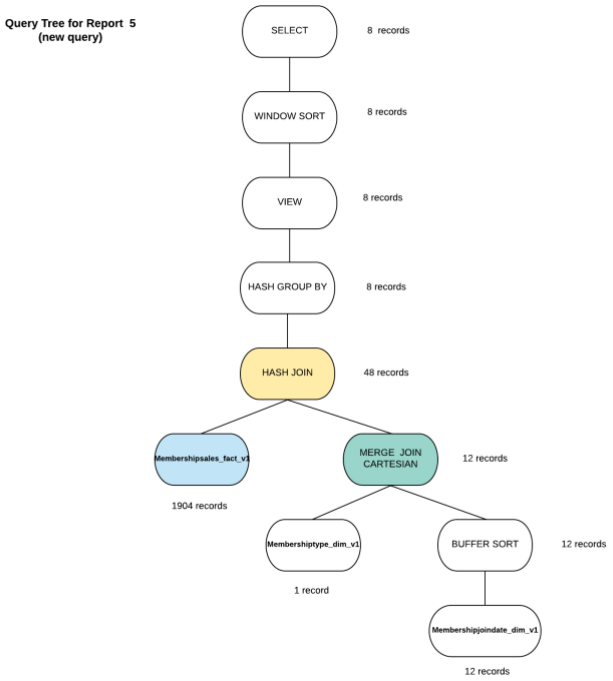
We find the result of new query is the same as original one. The result has 12 rows in total.

g. Execution plan of new query

```
Explained.

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 2158530460

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                            |     8 |   104 |    12   (9)| 00:00:01 |
|   1 |  WINDOW SORT             |                            |     8 |   104 |    12   (9)| 00:00:01 |
|   2 |   VIEW                   |                            |     8 |   104 |    12   (9)| 00:00:01 |
|   3 |    HASH GROUP BY         |                            |     8 |   312 |    12   (9)| 00:00:01 |
|*  4 |     HASH JOIN            |                            |    48 |  1872 |    11   (0)| 00:00:01 |
|   5 |      MERGE JOIN CARTESIAN|                            |    12 |   288 |     6   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
|*  6 |       TABLE ACCESS FULL  | MEMBERSHIPTYPE_DIM_V1      |     1 |     9 |     3   (0)| 00:00:01 |
|   7 |       BUFFER SORT        |                            |    12 |   180 |     3   (0)| 00:00:01 |
|*  8 |        TABLE ACCESS FULL | MEMBERSHIPJOINDATE_DIM_V1  |    12 |   180 |     3   (0)| 00:00:01 |
|   9 |      TABLE ACCESS FULL   | MEMBERSHIPSALES_FACT_V1    |  1904 | 28560 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("MS"."MEMBERSHIPTYPEID"="MT"."MEMBERSHIPTYPEID" AND
              "MS"."MEMBERSHIPJOINDATEID"="MD"."MEMBERSHIPJOINDATEID")

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
   6 - filter("MT"."MEMBERSHIPNAME"='Gold')
   8 - filter("MD"."MEMBERSHIPJOINYEAR"='2009')

24 rows selected.
```

From the result, we find number of records being processed has been reduced.

h. Query Tree of new query



From the query tree of new query, we find we change HASH JOIN into Cartesian product method. And then we reduce the number of records being processed.

i. Compare two query and find better one

Compared to original query, we find that number of records being processed in execution plan of new query is less. It means the processing time of new query is faster than original one. So, we think the **new query is more efficient than original query**.

Report 6

a. SQL for report 6

```
EXPLAIN PLAN FOR
SELECT
    dest_a.country AS country,
    dest_a.city AS city,
    SUM(total_number_of_routes) AS total_incoming_routes,
    RANK() OVER(
        PARTITION BY dest_a.country
        ORDER BY
            SUM(total_number_of_routes) DESC
    ) AS city_rank
FROM
    airport_dim_v1 dest_a,
    route_fact_v1 r
WHERE
    dest_a.airportid = r.destairportid
GROUP BY
    dest_a.country,
    dest_a.city;

SELECT * FROM TABLE ( dbms_xplan.display );
```

b. Screenshot of the query result

COUNTRY	CITY	TOTAL_INCOMING_ROUTES	CITY_RANK
Afghanistan	Kabul	23	1
Afghanistan	Herat	6	2
Afghanistan	Kandahar	4	3
Albania	Tirana	23	1
Algeria	Algier	57	1
Algeria	Oran	14	2
Algeria	Setif	11	3
Algeria	Annaba	10	4
Algeria	Constantine	8	5
Algeria	Tlemcen	5	6
Algeria	Batna	5	6

COUNTRY	CITY	TOTAL_INCOMING_ROUTES	CITY_RANK
Algeria	Bejaja	5	6
Algeria	Biskra	3	9
Algeria	Illizi	3	9
Algeria	Djanet	3	9
Algeria	Ghardaia	3	9
Algeria	Tamanrasset	3	9
Algeria	Ouargla	1	14
American Samoa	Pago Pago	2	1
Angola	Luanda	37	1
Angola	Lubango	6	2
Angola	Ondjiva	4	3

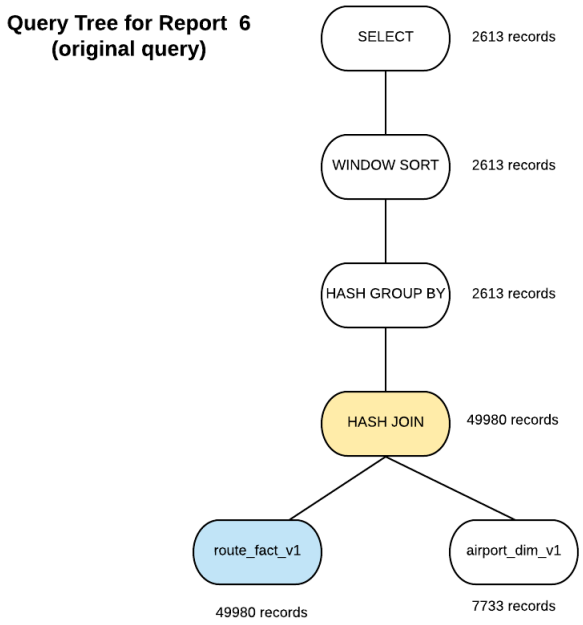
This is only part of query result. The result has 2613 rows in total.

c. Execution plan of the original query

```
Explained.

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------
Plan hash value: 1374591922

-------------------------------------------------------------------------------------------------
| Id  | Operation            | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                |  2613 | 81003 |   106   (5)| 00:00:01 |
|   1 |  WINDOW SORT         |                |  2613 | 81003 |   106   (5)| 00:00:01 |
|   2 |   HASH GROUP BY      |                |  2613 | 81003 |   106   (5)| 00:00:01 |
|*  3 |    HASH JOIN         |                | 49980 |  1513K|   102   (1)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| AIRPORT_DIM_V1 |  7733 |   181K|    17   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| ROUTE_FACT_V1  | 49980 |   341K|    84   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("DEST_A"."AIRPORTID"="R"."DESTAIRPORTID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - 1 Sql Plan Directive used for this statement

22 rows selected.
```

From the execution plan, we know there are 6 operations. Then we plot query tree to observe this execution plan.

d. Query Tree of the original query



From the query tree of original query, we find “WINDOW SORT” and “HASH GROUP BY” are in execution plan because of ranking query.

e. New Query

In this case, we think execution plan of original query is good. And we want to change them using method of optimizing cumulative and moving aggregate queries. Then finding the difference between original query and new one.

```
EXPLAIN PLAN FOR
SELECT
    innerquery.country,
    innerquery.city,
    innerquery.total_incoming_routes,
    RANK() OVER(
        PARTITION BY country
        ORDER BY
            total_incoming_routes DESC) AS city_rank
FROM
    ( (
        SELECT /*+ no_merge */
            dest_a.country AS country,
            dest_a.city AS city,
            SUM(total_number_of_routes) AS total_incoming_routes
        FROM
            airport_dim_v1 dest_a,
            route_fact_v1 r
        WHERE
            dest_a.airportid = r.destairportid
        GROUP BY
            dest_a.country,
            dest_a.city
    ) innerquery );

SELECT * FROM TABLE ( dbms_xplan.display );
```

f. Result of new query

COUNTRY	CITY	TOTAL_INCOMING_ROUTES	CITY_RANK
Afghanistan	Kabul	23	1
Afghanistan	Herat	6	2
Afghanistan	Kandahar	4	3
Albania	Tirana	23	1
Algeria	Algier	57	1
Algeria	Oran	14	2
Algeria	Setif	11	3
Algeria	Annaba	10	4
Algeria	Constantine	8	5
Algeria	Tlemcen	5	6
Algeria	Batna	5	6

COUNTRY	CITY	TOTAL_INCOMING_ROUTES	CITY_RANK
Algeria	Bejaja	5	6
Algeria	Biskra	3	9
Algeria	Illizi	3	9
Algeria	Djanet	3	9
Algeria	Ghardaia	3	9
Algeria	Tamanrasset	3	9
Algeria	Ouargla	1	14
American Samoa	Pago Pago	2	1
Angola	Luanda	37	1
Angola	Lubango	6	2
Angola	Ondjiva	4	3

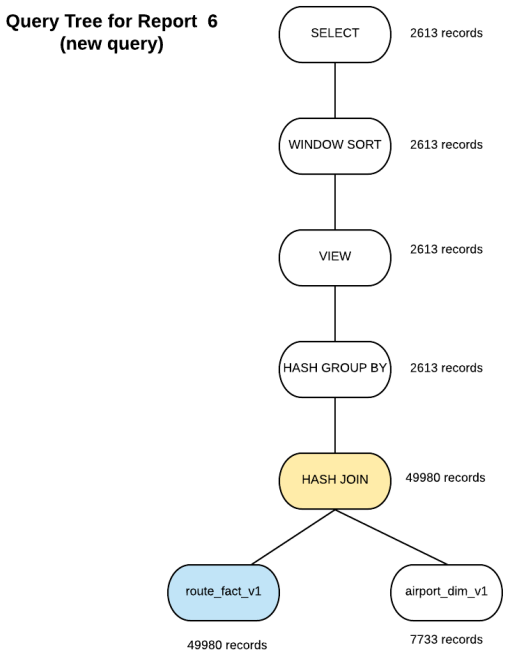
We find the result of new query is the same as original one. The result has 2613 rows in total.

g. Execution plan of new query

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
Plan hash value: 2174289798

-------------------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |  2613 |   127K|   105   (4)| 00:00:01 |
|   1 |  WINDOW SORT          |                |  2613 |   127K|   105   (4)| 00:00:01 |
|   2 |   VIEW                |                |  2613 |   127K|   104   (3)| 00:00:01 |
|   3 |    HASH GROUP BY      |                |  2613 | 81003 |   104   (3)| 00:00:01 |
|*  4 |     HASH JOIN         |                | 49980 |  1513K|   102   (1)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| AIRPORT_DIM_V1 |  7733 |   181K|    17   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
|   6 |      TABLE ACCESS FULL| ROUTE_FACT_V1  | 49980 |   341K|    84   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("DEST_A"."AIRPORTID"="R"."DESTAIRPORTID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
   - 1 Sql Plan Directive used for this statement

23 rows selected.
```

From the result, we find there are 7 operations which is more than original one.

h. Query Tree of new query



From the query tree of new query, we find total number of records processed are larger than original one.

i. Compare two query and find better one

Compared to original query, we find the execution plan of new query have one more operation “VIEW”. Then we find number of records processed of other part didn’t change but the number of operations has increased. Therefore, we know the **original query is more efficient than new query**.