Group Assignment

Open Flights and Travel

Implementing a data warehouse

Charul Kishor Manglani (26328046)

Jayesh Parab (27148572)

Simran Malik (27106721)

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
26328046	Kishor Manglani	Charul
27148572	Parab	Jayesh
27106721	Malik	Simran

***** Please include the names of all other group members.

Unit name and code	Database analysis and processing and FIT5137		
Title of assignment	Open flights and Travel Group assignment		
Lecturer/tutor	David Tania/Zhou Shao		
Tutorial day and time	Thursday and 12 noon	**Campus**	Caulfield
Is this an authorised group assignment? ☑ **Yes** ☐ **No**			
Has any part of this assignment been previously submitted as part of another unit/course? ☐ **Yes** ☑ **No**			
Due Date 17/10/2016		**Date submitted**	16/10/2016

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date) ..... Signature of lecturer/tutor .....

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - i. provide to another member of faculty and any external marker; and/or
  - ii. submit it to a text matching software; and/or
  - iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature .....Jayesh Parab..... *Date*…16/10/2016……

* delete (iii) if not applicable

Signature Charul Kishor Manglani Date: 16/10/2016 Signature _____ Date:_____

Signature Simran Malik Date: 16/10/2016 Signature _____ Date:_____

Signature _____ Date:_____ Signature _____ Date:_____

Privacy Statement

The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

Table of Contents

Details of Oracle Account used:

Username1: S27148572
Password1: student

Username2: S26328046
Password2: student

Username3: S27106721
Password3: student

For the assignment, we have used username 1 and username 2

Contribution Declaration Form

Percentage of contribution:

1. Name: Charul Kishor Manglani, ID: 26328046, Contribution: 33.34%
2. Name: Jayesh Parab, ID: 27148572, Contribution: 33.34%
3. Name: Simran Malik, ID: 27106721, Contribution: 33.34%

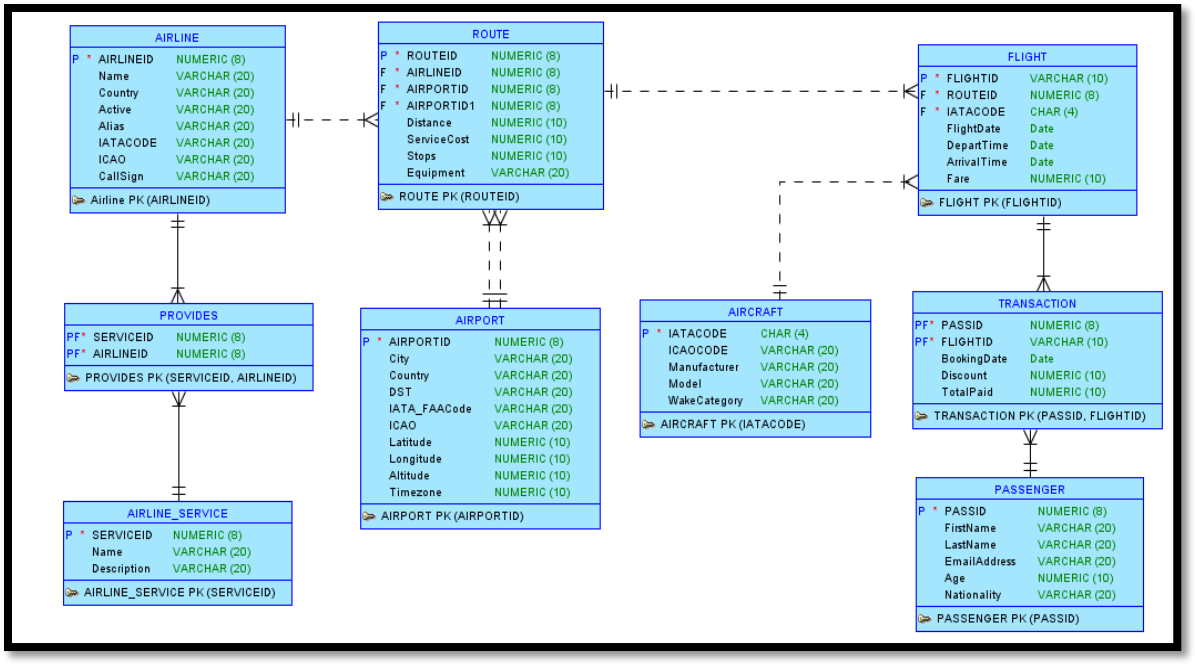
All the three group members have contributed equally to each part of the assignment.

1. Task C.1.

a) Copy the operational database from dw_oft into your account. Create the necessary primary keys and foreign keys for each table and draw E/R diagram using SQL Developer

```
create table Airport
as
select * from dw_oft.Airports1;

create table Airline
as
select * from dw_oft.Airlines1;

create table Route
as
select * from dw_oft.Routes1;

create table Aircraft
as
select * from dw_oft.Aircrafts1;

create table Provides
as
select * from dw_oft.Provides1;

create table Flight
as
select * from dw_oft.Flights1;

create table Passenger
as
select * from dw_oft.Passengers1;

create table Transaction
as
select * from dw_oft.Transactions1;

create table Airline_Service
as
select * from DW_OFT.AIRLINE_SERVICES1;
```

E/R Diagram with required primary keys and foreign keys



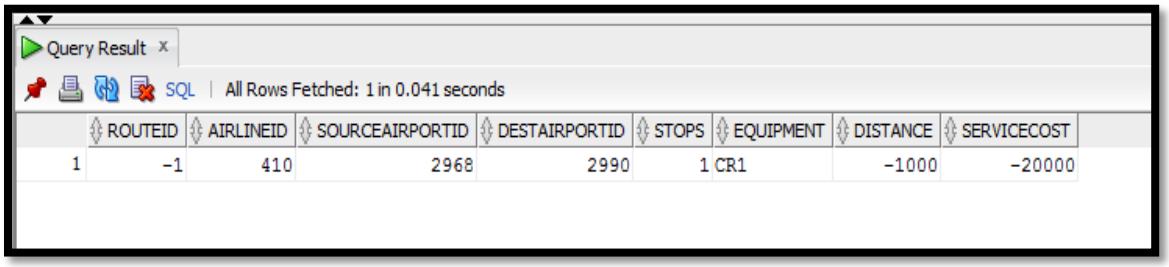
b) If you have done the data cleaning, explain what kind of data cleaning process that you have done (you need to show the SQL to explore the operational database, and SQL of the data cleaning, as well as the screenshot of data before and after data cleaning)

Doing data cleaning for Route table

```
--Total Records
Select count(*) from ROUTE;
--58445

-- Checking whether source airport id is null
Select COUNT(*) from ROUTE where SOURCEAIRPORTID is null;
--0

--Checking total number of source airport
Select COUNT(*) from ROUTE where SOURCEAIRPORTID is not null;
--58445

-- Checking whether destination airport id is null
Select COUNT(*) from ROUTE where DESTAIRPORTID is null;
--0

--Checking total number of destination airport
Select COUNT(*) from ROUTE where DESTAIRPORTID is not null;
--58445
```

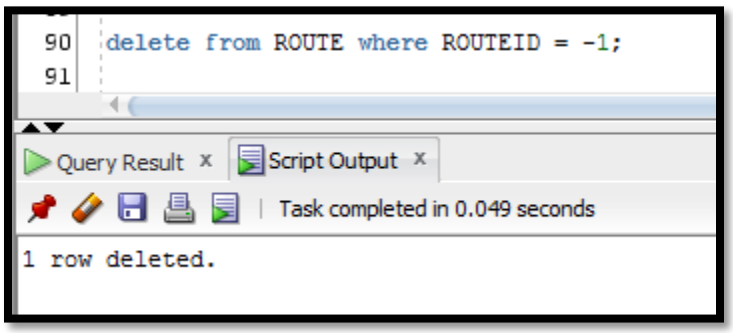
```
--Checking whether source and destination airport id are same
Select COUNT(*) from ROUTE where DESTAIRPORTID = SOURCEAIRPORTID;
--0

--Conclusion
--source airport id and destination airport id is not null in routes
--source airport id is not same as destination airport id in any row

--Checking for 0 distance travelled
Select COUNT(*) from ROUTE where distance = 0;
--0

--Checking for 0 service cost
Select COUNT(*) from ROUTE where SERVICECOST = 0;
--0

--Conclusion no distance and service cost is zero so value is present

-- Checking negative values for distance
Select * from ROUTE where DISTANCE <= 0;
-- Found one value with negative distance. We are deleting the record having routeid = -1

-- Checking negative values for servicecost
Select * from ROUTE where SERVICECOST <= 0;
-- Found one value with negative servicecost. We are deleting the record having routeid = -1

-- Checking negative values for routeid
Select * from ROUTE where ROUTEID <= 0;
-- Found one value with negative routeID. We are deleting the record having routeid = -1
```

Screenshot showing data before data cleaning

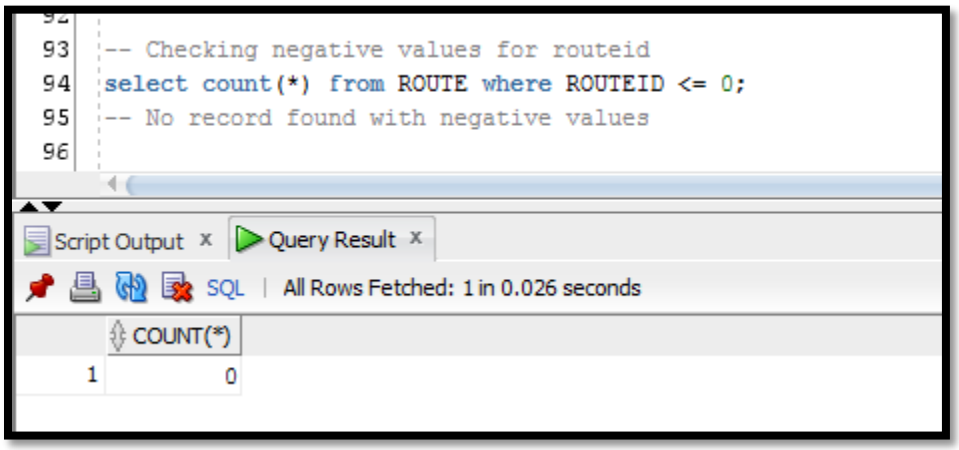


	ROUTEID	AIRLINEID	SOURCEAIRPORTID	DESTAIRPORTID	STOPS	EQUIPMENT	DISTANCE	SERVICECOST
1	-1	410	2968	2990	1	CR1	-1000	-20000

Deleting the record with routeid = -1 as distance, route ID, and service cost cannot be negative using the query shown below:



Screenshot after data cleaning to show that no record with negative value exists



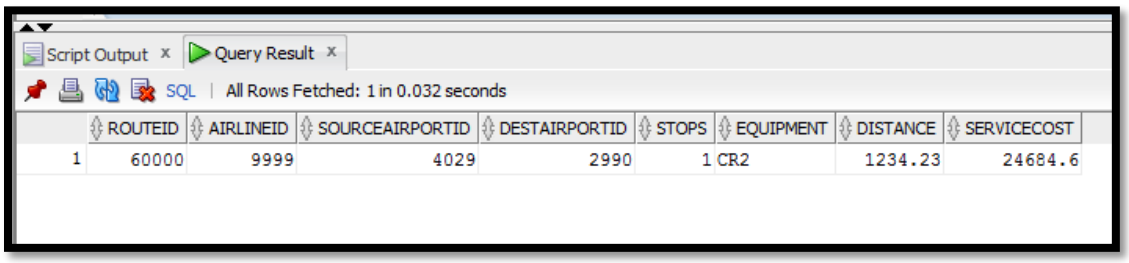
-- Checking whether any source airport id does not match airport id
Select count (*) from ROUTE where SOURCEAIRPORTID not IN (Select a.AIRPORTID from airport a);
--0

-- Checking whether any destination airport id does not match airport id
Select count (*) from ROUTE where DESTAIRPORTID not IN (Select a.AIRPORTID from airport a);
--0

--Checking whether airline id in route is matching airline id in airline table
Select count (*) from ROUTE where AIRLINEID not IN (Select a.AIRLINEID from AIRLINE a);
--1

-- fetching that 1 record
Select * from ROUTE where AIRLINEID not IN (Select a.AIRLINEID from AIRLINE a);

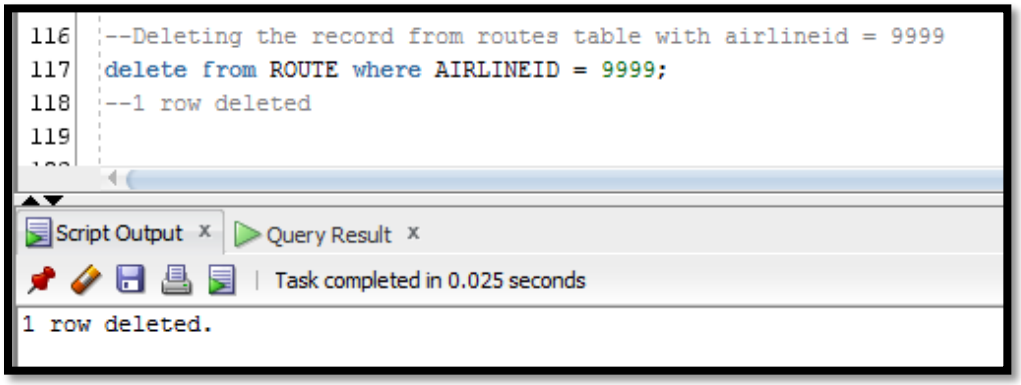
Screenshot showing that record with invalid airlineid



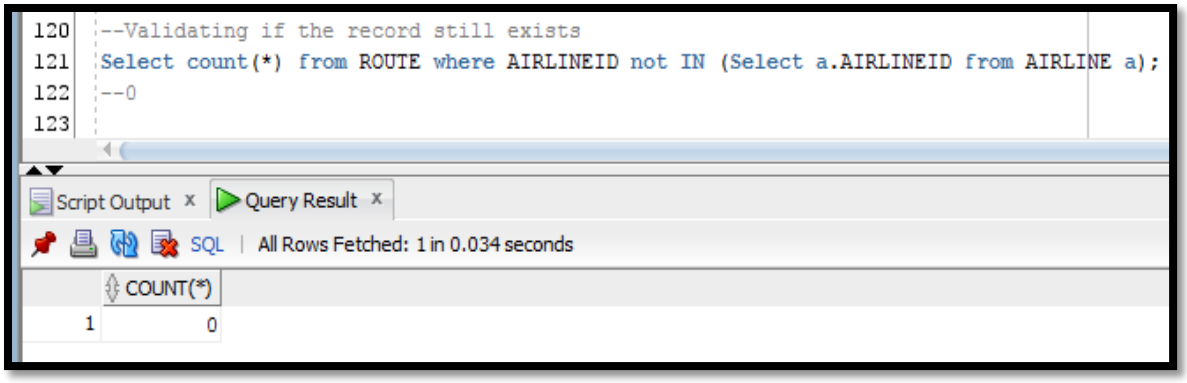
	ROUTEID	AIRLINEID	SOURCEAIRPORTID	DESTAIRPORTID	STOPS	EQUIPMENT	DISTANCE	SERVICECOST
1	60000	9999	4029	2990	1	CR2	1234.23	24684.6

--Validating the record with airlineid = 9999
Select count (*) from AIRLINE where AIRLINEID = 9999;
--0
--The row with airline id = 9999 does not exist so we delete this record as it is invalid

--Deleting the record from routes table with airlineid = 9999
Delete from ROUTE where AIRLINEID = 9999;
--1 row deleted



Screenshot showing that the invalid record does not exist now



--checking route id is unique
Select distinct count(*) from ROUTE;
--58443

Select count(*) from ROUTE;
--58443

--Checking whether there is any route is without airline
Select COUNT(*) from ROUTE where AIRLINEID is not null;
--58443

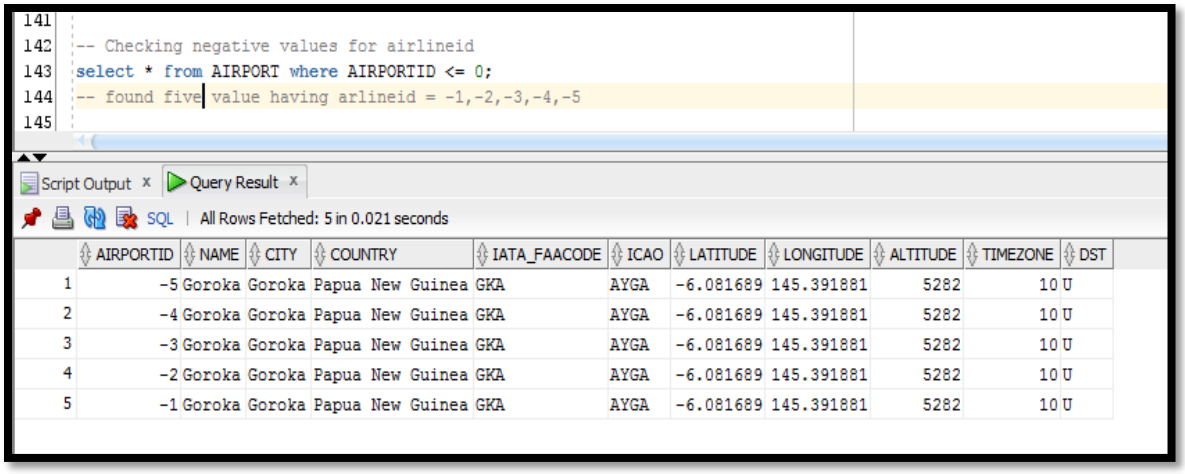
Doing data cleaning for airport table

Select COUNT(*) from AIRPORT where AIRPORTID is null;
--0

Select COUNT(*) from AIRPORT where AIRPORTID is not null;
--7738

-- Checking negative values for airlineid
select * from AIRPORT where AIRPORTID <= 0;
-- found five value having arlineid = -1,-2,-3,-4,-5

Screenshot showing that records with invalid AirportId:

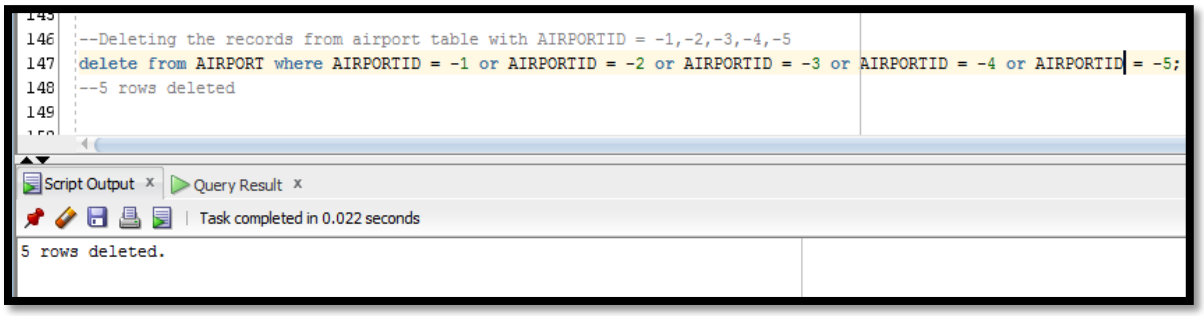


```
-- Checking negative values for airlineid
select * from AIRPORT where AIRPORTID <= 0;
-- found five value having arlineid = -1,-2,-3,-4,-5
```

Script Output | Query Result | SQL | All Rows Fetched: 5 in 0.021 seconds

	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

--Deleting the records from airport table with AIRPORTID = -1,-2,-3,-4,-5
delete from AIRPORT where AIRPORTID = -1 or AIRPORTID = -2 or AIRPORTID = -3 or AIRPORTID = -4 or AIRPORTID = -5;
--5 rows deleted

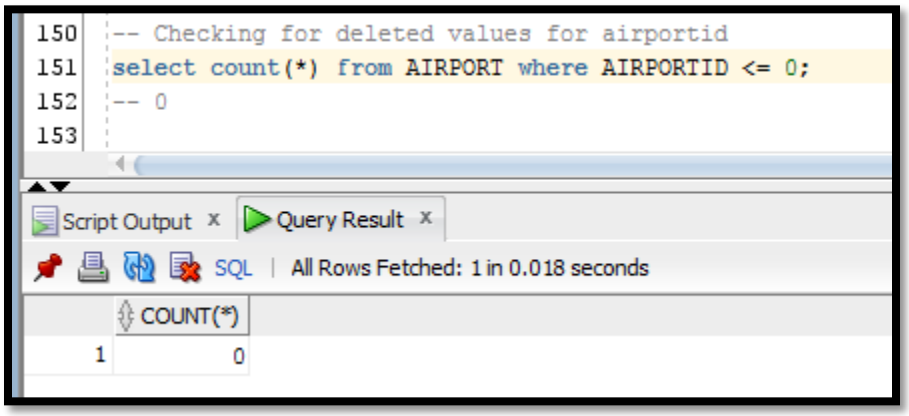


```
--Deleting the records from airport table with AIRPORTID = -1,-2,-3,-4,-5
delete from AIRPORT where AIRPORTID = -1 or AIRPORTID = -2 or AIRPORTID = -3 or AIRPORTID = -4 or AIRPORTID = -5;
--5 rows deleted
```

Script Output | Query Result | Task completed in 0.022 seconds

```
5 rows deleted.
```

Screenshot showing that the invalid record does not exist now



```
-- Checking for deleted values for airportid
select count(*) from AIRPORT where AIRPORTID <= 0;
-- 0
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.018 seconds

	COUNT(*)
1	0

Doing data cleaning for Airline service table

--Checking airline_service

```
select * from AIRLINE_SERVICE;
-- 11

-- checking whether there is any null values
Select COUNT(*) from AIRLINE_SERVICE where SERVICEID is  null;
-- 0

-- checking for any Repetitions is any name values
select distinct count(name) from AIRLINE_SERVICE;
-- 11

-- Checking negative values for serviceid
select count(*) from AIRLINE_SERVICE where SERVICEID <= 0;
-- 0
```

Doing data cleaning for airline table

```
-- Checking Airline

-- Checking for number of records in airline
Select COUNT(*) from AIRLINE;
--5987 records

-- Checking for null values in AirlineID
Select COUNT(*) from AIRLINE where AIRLINEID is null;
-- 0

-- Checking negative values for airlineid
select * from AIRLINE where AIRLINEID <= 0;
-- found one value having arlineid = -1
```

Screenshot showing record with airlineID = -1

```
181 -- Checking negative values for airlineid
182 select * from AIRLINE where AIRLINEID <= 0;
183 -- found one value having arlineid = -1
184
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.032 seconds

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

-- deleting one record having airlineID = -1
Delete from AIRLINE where AIRLINEID = -1;
-- 1 row deleted.

Screenshot showing the deletion of invalid record

```
184
185 -- deleting one record having airlineID = -1
186 Delete from AIRLINE where AIRLINEID = -1;
187 -- 1 row deleted.
188
```

Script Output | Query Result | Task completed in 0.021 seconds

```
1 row deleted.
```

Screenshot showing that the invalid records don't exist anymore.

```
189 -- Verifying the records having a negative airlineId no more exists
190 select count(*) from AIRLINE where AIRLINEID <= 0;
191 -- 0
192
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.023 seconds

	COUNT(*)
1	0

Doing data cleaning for provides table

--checking provides

-- Checking for the number of records in Provides
select count(*) from PROVIDES;
-- 20981

--Checking for any unmatched airline and service IDs from airline and airline_service
Select count(*) from PROVIDES where AIRLINEID not IN (Select a.AIRLINEID
from AIRLINE a);
-- 0

-- Checking for any serviceID that does not exist in Airline_Service table
Select count(*) from PROVIDES where SERVICEID not IN (Select s.SERVICEID
from AIRLINE_SERVICE s);
-- 0

-- Checking negative values for serviceid
select count(*) from PROVIDES where SERVICEID <= 0;
-- found 0 value having ServiceID

-- Checking negative values for airlineID
select count(*) from PROVIDES where AIRLINEID <= 0;
-- found 0 value having AirlineID

Doing data cleaning for flights table

-- Checking Flights
select count(*) from FLIGHT;
--50002

--checking for null values
select count(*) from FLIGHT where ROUTEID is null;
--0

select count(*) from FLIGHT where AIRCRAFTID is null;
--0

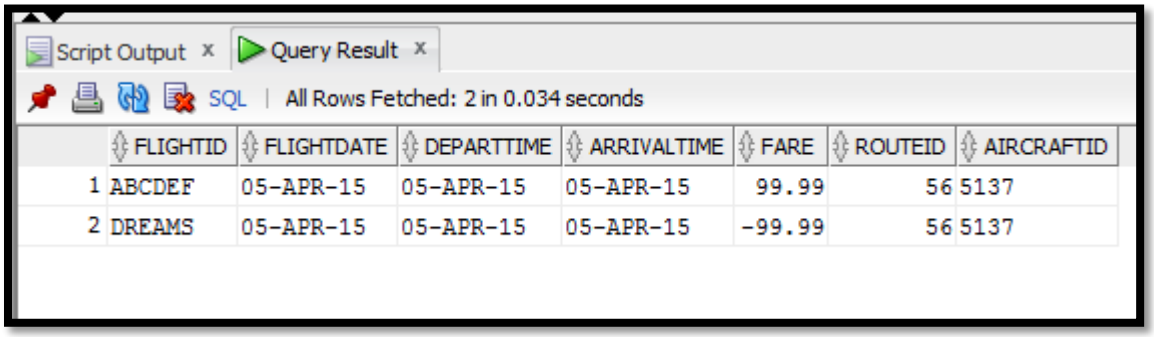
select count(*) from FLIGHT where FLIGHTID is null;
--0

-- Checking for unmatched values for routeid
Select count(*) from FLIGHT where ROUTEID not IN (Select r.ROUTEID from ROUTE r);
--0

-- Checking for unmatched values for aircraftID
Select count(*) from FLIGHT where AIRCRAFTID not IN (Select a.IATACODE from AIRCRAFT a);
--2

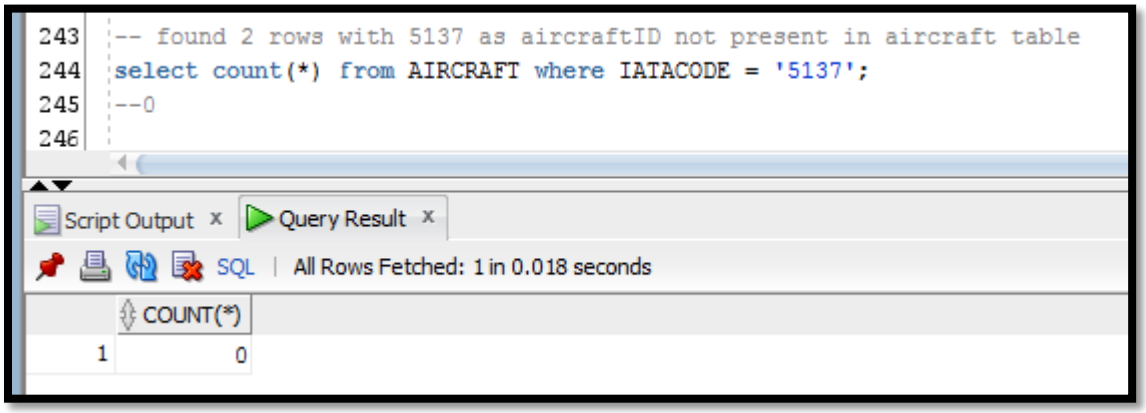
--fetching 2 invalid rows
Select * from FLIGHT where AIRCRAFTID not IN (Select a.IATACODE from AIRCRAFT a);
--We are deleting these two rows as they have invalid aircraftIDs

Screenshot showing records with invalid aircraftID values



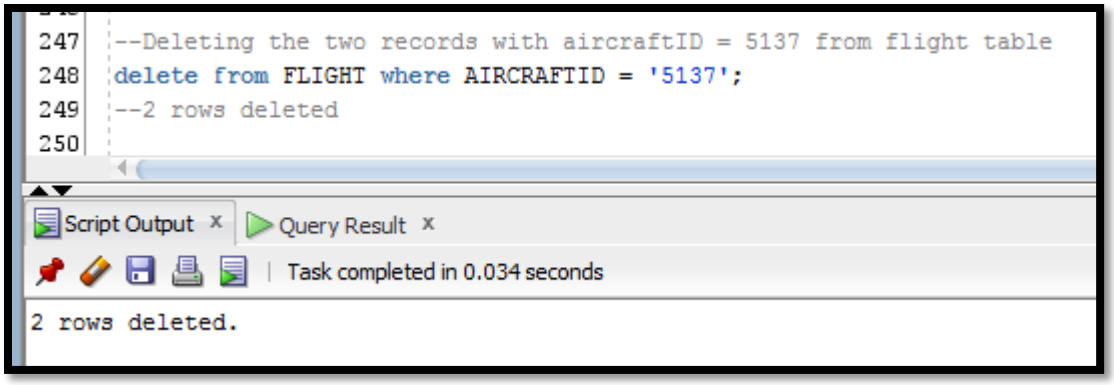
	FLIGHTID	FLIGHTDATE	DEPARTTIME	ARRIVALTIME	FARE	ROUTEID	AIRCRAFTID
1	ABCDEF	05-APR-15	05-APR-15	05-APR-15	99.99	56	5137
2	DREAMS	05-APR-15	05-APR-15	05-APR-15	-99.99	56	5137

Validating if the above records with aircraftid = 5137 exists in aircraft table:

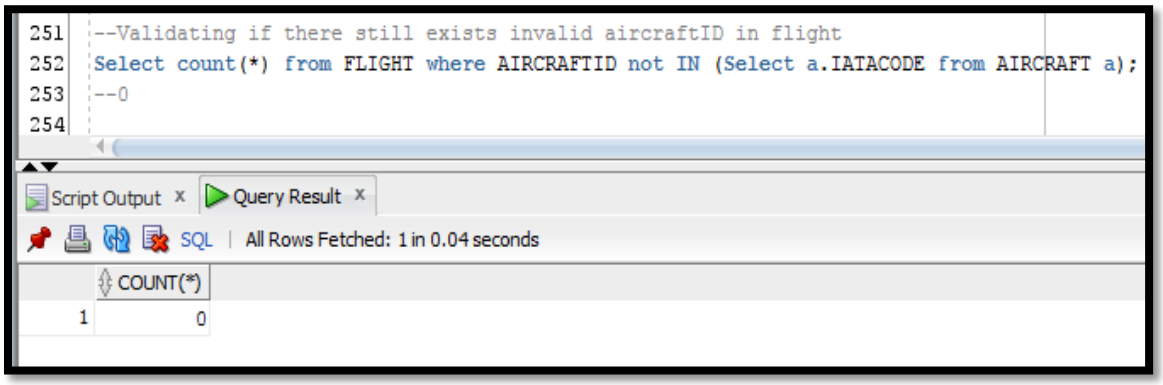


	COUNT(*)
1	0

Deleting these invalid records from flight table



Screenshot showing that there are no more invalid records in the flight table



--checking for negative fares
Select count(*) from FLIGHT where fare <= 0;
--0

-- Checking invalid values for FlightID
Select count(*) from FLIGHT where FLIGHTID LIKE '-%';
-- 0

--Checking for negative values for routeid
Select count(*) from FLIGHT where ROUTEID <= 0;
--0

--Checking for invalid values for aircraftid
Select count(*) from FLIGHT where AIRCRAFTID LIKE '-%';
-- 0

Doing data cleaning for aircraft table

--Checking aircraft
select count(*) from AIRCRAFT;

```
-- 357

--checking for negative values
select count(*) from AIRCRAFT where IATACODE like '-%';
-- 0

--checking for invalid wakecategory
select * from AIRCRAFT where WAKECATEGORY not in ('M','H','L');
-- 4 rows
-- Confused if we should delete these
-- Some rows are with model as null.. will it matter?

select * from AIRCRAFT where MODEL LIKE '-%';
--We found 5 rows with – in the front
```

We are neither deleting nor updating as it will not affect our data warehouse.

```
select count(*) from AIRCRAFT where MANUFACTURER LIKE '-%';
--found zero rows
```

Doing data cleaning for passenger table

```
--Checking passenger
select count(*) from PASSENGER;
-- 10004 records found but assignment specification tells that we have records of 10000
passengers

-- Checking for negative values of PASSID
select * from PASSENGER where PASSID <= 0;
-- found 0 rows

-- Checking for negative values in Age
select count(*) from PASSENGER where AGE < 0;
-- 5 rows

--fetching 5 records
select * from PASSENGER where AGE < 0;
-- we are deleting these records as age cannot be negative.
```

Screenshot showing records with negative age values

```
311 --fetching 5 records
312 select * from PASSENGER where AGE < 0;
313 -- we are deleting these records as age cannot be negative.
314
```

Script Output | Query Result | SQL | All Rows Fetched: 5 in 0.032 seconds

	PASSID	FIRSTNAME	LASTNAME	EMAILADDRESS	AGE	NATIONALITY
1	1000	Wayne	Rooney	wayne@Gmail.com	-1	English
2	999	Juan	Mata	juan@Gmail.com	-1	Spainish
3	998	Adan	Januzaj	adan@Gmail.com	-1	Belgian
4	997	Luke	Shaw	luke@Gmail.com	-1	English
5	996	Phil	Jones	phil@Gmail.com	-1	English

Deleting these records as age cannot be negative

```
315 --Deleting 5 records
316 Delete from PASSENGER where age < 0;
317 -- 5 rows deleted
318
```

Script Output | Query Result | Task completed in 0.025 seconds

```
5 rows deleted.
```

Screenshot after the records with negative age value are deleted

```
319 --Validating if any other records with negative age value exits
320 select count(*) from PASSENGER where AGE < 0;
321 --0
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.115 seconds

	COUNT(*)
1	0

-- Checking for 0 values in Age
select count(*) from PASSENGER where AGE = 0;
-- found 149 rows but we are keeping these values as infant can also travel

Doing data cleaning for transaction table

-- Checking for values
select count(*) from TRANSACTION;
-- 25005 records found.

-- Checking for unmatched PassId values
Select count(*) from TRANSACTION where PASSID not IN (Select p.PASSID from PASSENGER p);
-- found 0 rows.

-- Checking for unmatched FlightID values
Select count(*) from TRANSACTION where FLIGHTID not IN (Select f.FLIGHTID from FLIGHT f);
-- 5 values found

Screenshot showing invalid records in transaction table

```
343 --fetching those 5 records
344 Select * from TRANSACTION where FLIGHTID not IN (Select f.FLIGHTID
345 from FLIGHT f);
346 -- we are deleting these 5 values
347
```

Script Output | Query Result | SQL | All Rows Fetched: 5 in 0.059 seconds

	PASSID	FLIGHTID	BOOKINGDATE	DISCOUNT	TOTALPAID
1	1001	GHOST5	05-APR-15	0	999
2	1001	GHOST1	05-APR-15	0	999
3	1001	GHOST4	05-APR-15	0	999
4	1001	GHOST2	05-APR-15	0	999
5	1001	GHOST3	05-APR-15	0	999

--Checking for these values in flight table
select * from FLIGHT where FLIGHTID like 'GHOST%';
--0

Screenshot showing the deletion of the invalid records

```
352  --Deleting these 5 records
353  delete from TRANSACTION where FLIGHTID like 'GHOST%';
354  --5 rows deleted
```

Script Output | Query Result | Task completed in 0.034 seconds

```
5 rows deleted.
```

Screenshot showing that the transaction table does not have any records with invalid flightID values

```
356  --Validating if the transaction table is now cleaned
357  Select count(*) from TRANSACTION where FLIGHTID not IN (Select f.FLIGHTID
358  from FLIGHT f);
359  --0
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.051 seconds

	COUNT(*)
1	0

-- Checking for negative values in Totalpaid
select count(*) from TRANSACTION where TOTALPAID <= 0;
-- found 0 rows

-- Checking for negative values in Discount ????
select count(*) from TRANSACTION where DISCOUNT < 0;
-- found 0 rows

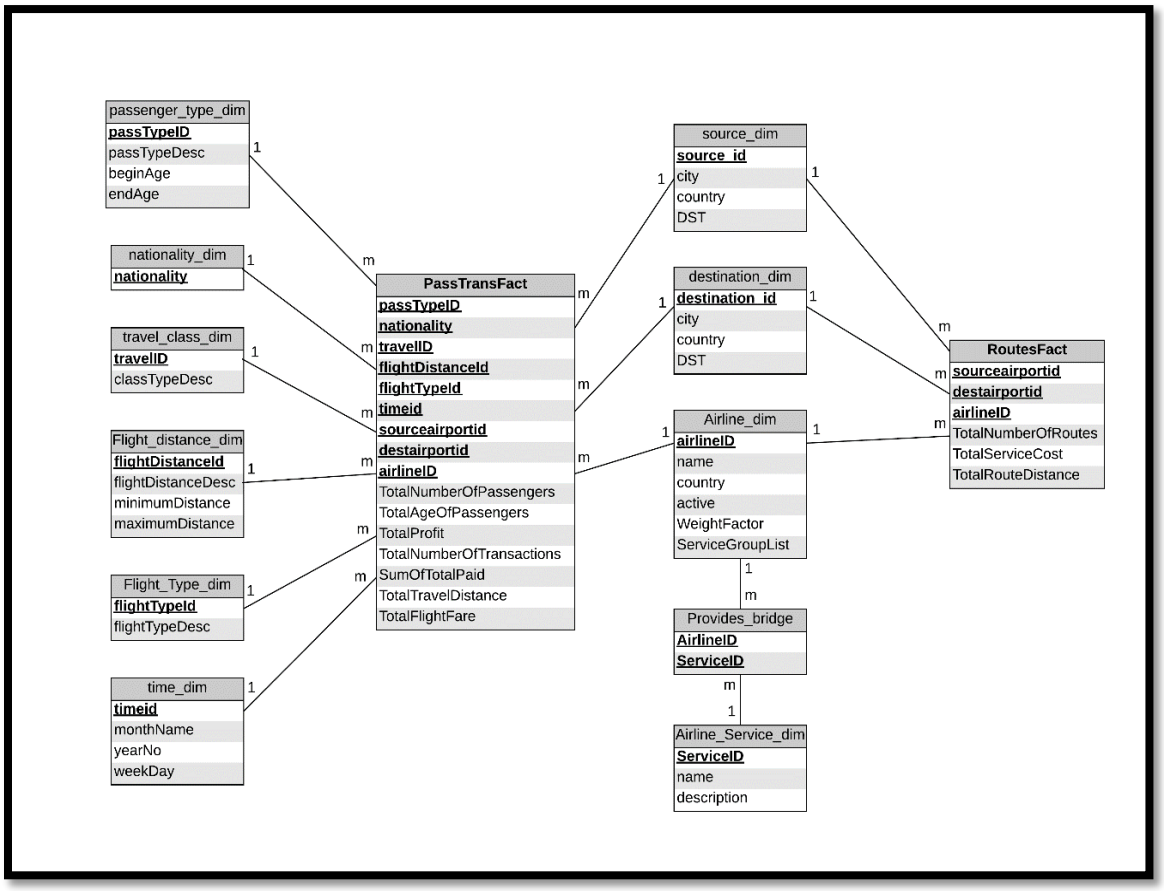
-- Checking for booking date between 2006-09
select count(*) from TRANSACTION where to_char(BOOKINGDATE,'YYYY') < '2006'
OR to_char(BOOKINGDATE,'YYYY') > '2009';
-- 0 rows

Queries to select the data from the tables:

select * from AIRLINE;
Select * from AIRCRAFT;
Select * from ROUTE;
Select * from Airport;

```
select * from PASSENGER;
select * from TRANSACTION;
select * from PROVIDES;
select * from AIRLINE_SERVICE;
select * from FLIGHT;
```

c) A star/snowflake schema (diagram) for your data warehouse



d) A short explanation of your star schema

The three main components of the above shown multi-fact snowflake schema are:

Facts

Facts are numeric measurements and are aggregate values. We reviewed the requirements of the management and found that some of the dimensions are not common to both the fact measures. Hence, we decided to create two fact tables as follows:

- PassTransFact: It contains measures related to passengers and flight transactions. It includes the following measures:

- TotalNumberOfPassengers
- TotalAgeOfPassengers
- TotalProfit (Total paid – flight fare)
- TotalNumberOfTransactions
- SumOfTotalPaid

- TotalTravelDistance
- TotalFlightFare
    - RoutesFact: It contains measures related to routes which are as follows:
- TotalNumberOfRoutes
- TotalServiceCost
- TotalRouteDistance

Dimensions

Dimensions are qualifying characteristics that provide additional perspectives to a given fact. Considering management's requirements, we have identified ten dimensions, out of which three dimensions are common to both fact tables. One of these common dimensions is connected to another dimension table through a bridge table because

- There is no direct relationship between routes and airline services
- The airline service information is not available in the service cost of a particular route

Dimensions related to PassTransFact only are:

- Passenger_type_dim
- Nationality_dim
- travel_Class_dim
- flight_distance_dim
- flight_type_dim
- time_dim

Dimensions related to both PassTransFact and RoutesFact are:

- source_dim
- destination_dim
- airline_dim
- airline_service_dim

There is no such dimension which is unique to RoutesFact.

Airline_dim dimension is connected to airline_service_dim dimension through a bridge table called provides_bridge and is linked through airlineid and serviceid with airline_dim and airline_service_dim respectively.

Attributes

These are the additional description of the dimension table and are optional. For example, the attributes for airline_dim are:

- AirlineID
- Name
- Country
- Active

We have also added WeightFactor and ServiceGroupList in airline_dim to depict the share of each service provided by that airline in the total service cost and the services which are available for that airline respectively.

e) SQL statements (e.g. create table, insert into, etc) to create the star schema

```
--Creating Source Dimension
Create table source_dim as
Select distinct a.AIRPORTID as source_id, a.city, a.country, a.DST
from AIRPORT a, ROUTE r
where a.AIRPORTID = r.SOURCEAIRPORTID
order by a.AIRPORTID;

--Creating Destination Dimension
Create table destination_dim as
Select distinct a.AIRPORTID as destination_id, a.city, a.country,a.DST
from AIRPORT a, ROUTE r
where a.AIRPORTID = r.DESTAIRPORTID
order by a.AIRPORTID;

--Creating Passenger_Type Dimension (children,Teenager,Young, Adult)
Create table passenger_type_dim
(passTypeID number,
 passTypeDesc varchar2 (20),
 beginAge number,
 endAge number
 );

Select max(age) from PASSENGER;
--87

--Inserting values into passenger type dimension
Insert into passenger_type_dim values (1, 'Children', 0, 10);
Insert into passenger_type_dim values (2, 'Teenager', 11, 17);
Insert into passenger_type_dim values (3, 'Young Adult', 18, 35);
Insert into passenger_type_dim values (4, 'Middle Adult', 36, 60);
--Add a assumption for max age 87 so we hav taken 99
Insert into passenger_type_dim values (5, 'Senior Adult', 61, 99);

--Creating Nationality Dimension
Create table nationality_dim as
Select distinct p.NATIONALITY
from PASSENGER p;
```

```
--Create travel class Dimension
Create table travel_class_dim
(travelID number,
 classTypeDesc varchar2 (20)
 );

-- Inserting values into travel class dimension
Insert into travel_class_dim values (1, 'Business Class');
Insert into travel_class_dim values (2, 'First Class');
Insert into travel_class_dim values (3, 'Economy Class');

--Create flight distance dimension
Create table Flight_distance_dim
(flightDistanceId number,
 flightDistanceDesc varchar2 (20),
 minimumDistance number,
 maximumDistance number
 );

--Inserting data into flight Distance dimension
Insert into Flight_distance_dim values (1, 'Small', 0,1199);
Insert into Flight_distance_dim values (2, 'Medium', 1200,4000);
Insert into Flight_distance_dim values (3, 'Large', 4001,10000);
Insert into Flight_distance_dim values (4, 'Very Large', 10001,19999);

--Create Airline dimension
Create Table Airline_dim As
Select a.AIRLINEID, a.NAME, a.COUNTRY, a.ACTIVE,
round(1.0/count(p.SERVICEID),2) As
WeightFactor , LISTAGG (p.SERVICEID, '_') Within Group (Order By
p.SERVICEID) As ServiceGroupList
From Airline a, Provides p
Where a.AIRLINEID = p.AIRLINEID
Group By a.AIRLINEID, a.NAME, a.COUNTRY, a.ACTIVE;

--Create flight_type dimension
Create table Flight_Type_dim
(flightTypeId number,
 flightTypeDesc varchar2 (20)
 );

--Insert data into flight type dimension
Insert into Flight_Type_dim values (1, 'Domestic');
```

```
Insert into Flight_Type_dim values (2, 'International');

--Creating Airline service dimension
Create table Airline_Service_dim
AS
Select * from AIRLINE_SERVICE;

--Creating provides bridge table
Create table Provides_bridge
AS
Select * from PROVIDES;

--Create time dimension
Create table time_dim as
Select distinct to_char(Flightdate,'MM') || to_char(Flightdate, 'YYYY') || to_char(Flightdate,
'DAY')as timeId, to_char(Flightdate, 'YYYY') as yearNo,
to_char(Flightdate,'MM') as monthName,
to_char(Flightdate, 'DAY') as weekDay from flight
order by yearNo;

--Create temporary passtransfact table
Create table temp_passtransfact as
Select r.sourceairportid, r.DESTAIRPORTID, p.nationality, ar.airlineid, count(p.passid) as
TotalNumberOfPassengers,
  sum(p.age) as TotalAgeOfPassengers, count(t.passid) as TotalNumberOfTransactions,
sum(t.totalpaid) as SumOfTotalPaid,
  sum(r.Distance) as TotalTravelDistance, sum(f.fare) as TotalFlightFare, p.AGE, f.fare,
t.TOTALPAID, r.DISTANCE, a1.COUNTRY as SourceCountry,
  a2.country as DestCountry, f.FLIGHTDATE
from Route r, airport a1, airport a2, flight f, transaction t, passenger p, airline ar
where r.sourceairportid = a1.AIRPORTID and
  r.DESTAIRPORTID = a2.AIRPORTID and
  r.routeid = f.routeid and f.flightid = t.flightid and
  p.passid = t.passid and
  ar.AIRLINEID = r.AIRLINEID
group by r.sourceairportid, r.DESTAIRPORTID, p.nationality, ar.airlineid, p.AGE, f.FARE,
t.TOTALPAID, r.distance, a1.COUNTRY, a2.country, f.FLIGHTDATE;


--Alter for passenger type
Alter table temp_passtransfact
add(passTypeID number);
```

```
--update for passenger type
update temp_passtransfact
set passTypeID =1
where Age >= 0
and Age <= 10;

update temp_passtransfact
set passTypeID =2
where Age >= 11
and Age <= 17;

update temp_passtransfact
set passTypeID =3
where Age >= 18
and Age <= 35;

update temp_passtransfact
set passTypeID =4
where Age >= 36
and Age <= 60;

update temp_passtransfact
set passTypeID =5
where Age >= 61
and Age <= 99;

-- Alter travel class dimension
Alter table temp_passtransfact
add(travelID number);

-- Update queries for travel class dimension

update temp_passtransfact
set TRAVELID = 1
where TOTALPAID >= 1.8*FARE;

update temp_passtransfact
set TRAVELID = 2
where TOTALPAID >= 1.3*FARE
and TOTALPAID < 1.8*FARE;

update temp_passtransfact
set TRAVELID = 3
```

```
where TOTALPAID < 1.3*FARE;

-- Alter table for Flight Distance Dim
Alter table temp_passtransfact
add(flightDistanceid number);

-- Update queries for Flight Distance Dim

update temp_passtransfact
set FLIGHTDISTANCEID = 1
where DISTANCE >= 0
AND DISTANCE <= 1199;

update temp_passtransfact
set FLIGHTDISTANCEID = 2
where DISTANCE >= 1200
AND DISTANCE <= 4000;

update temp_passtransfact
set FLIGHTDISTANCEID = 3
where DISTANCE >= 4001
AND DISTANCE <= 10000;

update temp_passtransfact
set FLIGHTDISTANCEID = 4
where DISTANCE >= 10001
AND DISTANCE <= 19999;

-- Alter table for Flight Type Dim
Alter table temp_passtransfact
add(flightTypeid number);

-- Update queries for Flight Type Dim

update temp_passtransfact
set flightTypeid = 1
where SOURCECOUNTRY = DESTCOUNTRY;

update temp_passtransfact
set flightTypeid = 2
where SOURCECOUNTRY != DESTCOUNTRY;

Create table passtransfact as
```

```
Select f.passtypeid, f.NATIONALITY, f.TRAVELID, f.FLIGHTDISTANCEID,
f.FLIGHTTYPEID,
      to_char(f.Flightdate,'MM') || to_char(f.Flightdate, 'YYYY') || to_char(f.Flightdate,
'DAY')as timeId,
      f.SOURCEAIRPORTID, f.DESTAIRPORTID, f.AIRLINEID,
f.TOTALNUMBEROFPASSENGERS, f.TOTALAGEOFPASSENGERS,
      sum (f.SUMOFTOTALPAID - f.TOTALFLIGHTFARE) as TotalProfit,
f.TOTALNUMBEROFTRANSACTIONS, f.SUMOFTOTALPAID,
      f.TOTALTRAVELDISTANCE, f.TOTALFLIGHTFARE
from temp_passtransfact f
group by f.passtypeid, f.NATIONALITY, f.TRAVELID, f.FLIGHTDISTANCEID,
f.FLIGHTTYPEID,
      to_char(f.Flightdate,'MM') || to_char(f.Flightdate, 'YYYY') || to_char(f.Flightdate,
'DAY'),
      f.SOURCEAIRPORTID, f.DESTAIRPORTID, f.AIRLINEID,
f.TOTALNUMBEROFPASSENGERS, f.TOTALAGEOFPASSENGERS,
      f.TOTALNUMBEROFTRANSACTIONS, f.SUMOFTOTALPAID,
      f.TOTALTRAVELDISTANCE, f.TOTALFLIGHTFARE;

--Fact Table 2

Create table Routesfact as
Select r.sourceairportid, r.DESTAIRPORTID, ar.airlineid, count(r.ROUTEID) as
TotalNumberOfRoutes,
  sum(r.SERVICECOST) as TotalServiceCost, sum (r.DISTANCE) as TotalRouteDistance
from Route r, airport a1, airport a2, airline ar
where r.sourceairportid = a1.AIRPORTID and
  r.DESTAIRPORTID = a2.AIRPORTID and
  ar.AIRLINEID = r.AIRLINEID
group by r.sourceairportid, r.DESTAIRPORTID, ar.airlineid;
```

f) Screen shots of the tables that you have created; this includes the contents of each table that you have created. If the table is very big, you can print a snapshot of the contents of the table

Screenshot showing passenger_type_dim table

```
611  --Queries to show the data of each created table
612  select * from PASSENGER_TYPE_DIM;
613
```

Script Output | Query Result | SQL | All Rows Fetched: 5 in 0.034 seconds

	PASSTYPEID	PASSTYPEDESC	BEGINAGE	ENDAGE
1	1	Children	0	10
2	2	Teenager	11	17
3	3	Young Adult	18	35
4	4	Middle Adult	36	60
5	5	Senior Adult	61	99

Screenshot showing nationality_dim table

```
613  select * from NATIONALITY_DIM;
614
615
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.041 seconds

	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

Screenshot showing travel_class_dim table

```
614 select * from TRAVEL_CLASS_DIM;
615
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.021 seconds

	TRAVELID	CLASSTYPEDESC
1	1	Business Class
2	2	First Class
3	3	Economy Class

Screenshot showing flight_distance_dim table

```
615 select * from FLIGHT_DISTANCE_DIM;
616
```

Script Output | Query Result | SQL | All Rows Fetched: 4 in 0.034 seconds

	FLIGHTDISTANCEID	FLIGHTDISTANCEDESC	MINIMUMDISTANCE	MAXIMUMDISTANCE
1	1	Small	0	1199
2	2	Medium	1200	4000
3	3	Large	4001	10000
4	4	Very Large	10001	19999

Screenshot showing flight_type_dim table

```
616 select * from FLIGHT_TYPE_DIM;
617
```

Script Output | Query Result | SQL | All Rows Fetched: 2 in 0.049 seconds

	FLIGHTTYPEID	FLIGHTTYPEDESC
1	1	Domestic
2	2	International

Screenshot showing time_dim table

```
617  select * from TIME_DIM;
```

Script Output × Query Result ×

SQL | Fetched 50 rows in 0.036 seconds

	TIMEID	YEARNO	MONTHNAME	WEEKDAY
1	012007FRIDAY	2007	01	FRIDAY
2	012007MONDAY	2007	01	MONDAY
3	012007SATURDAY	2007	01	SATURDAY
4	012007SUNDAY	2007	01	SUNDAY
5	012007THURSDAY	2007	01	THURSDAY
6	012007TUESDAY	2007	01	TUESDAY
7	012007WEDNESDAY	2007	01	WEDNESDAY
8	022007FRIDAY	2007	02	FRIDAY
9	022007MONDAY	2007	02	MONDAY
10	022007SATURDAY	2007	02	SATURDAY
11	022007SUNDAY	2007	02	SUNDAY
12	022007THURSDAY	2007	02	THURSDAY
13	022007TUESDAY	2007	02	TUESDAY
14	022007WEDNESDAY	2007	02	WEDNESDAY
15	032007FRIDAY	2007	03	FRIDAY
16	032007MONDAY	2007	03	MONDAY

Screenshot showing source_dim table

```
618  select * from SOURCE_DIM;
```

Script Output × Query Result ×

SQL | Fetched 50 rows in 0.026 seconds

	SOURCE_ID	CITY	COUNTRY	DST
1	1	Goroka	Papua New Guinea	U
2	2	Madang	Papua New Guinea	U
3	3	Mount Hagen	Papua New Guinea	U
4	4	Nadzab	Papua New Guinea	U
5	5	Port Moresby	Papua New Guinea	U
6	6	Wewak	Papua New Guinea	U
7	7	Narssarssuaq	Greenland	E
8	8	Godthaab	Greenland	U
9	9	Sondrestrom	Greenland	E
10	10	Thule	Greenland	E
11	11	Akureyri	Iceland	N

Screenshot showing destination_dim table

```
619  select * from DESTINATION_DIM;
620
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.038 seconds

	DESTINATION_ID	CITY	COUNTRY	DST
1	1	Goroka	Papua New Guinea	U
2	2	Madang	Papua New Guinea	U
3	3	Mount Hagen	Papua New Guinea	U
4	4	Nadzab	Papua New Guinea	U
5	5	Port Moresby	Papua New Guinea	U
6	6	Wewak	Papua New Guinea	U
7	7	Narssarssuaq	Greenland	E
8	8	Godthaab	Greenland	U
9	9	Sondrestrom	Greenland	E
10	10	Thule	Greenland	E
11	11	Akureyri	Iceland	N
12	12	Egilsstadir	Iceland	N
13	15	Isafjordur	Iceland	N

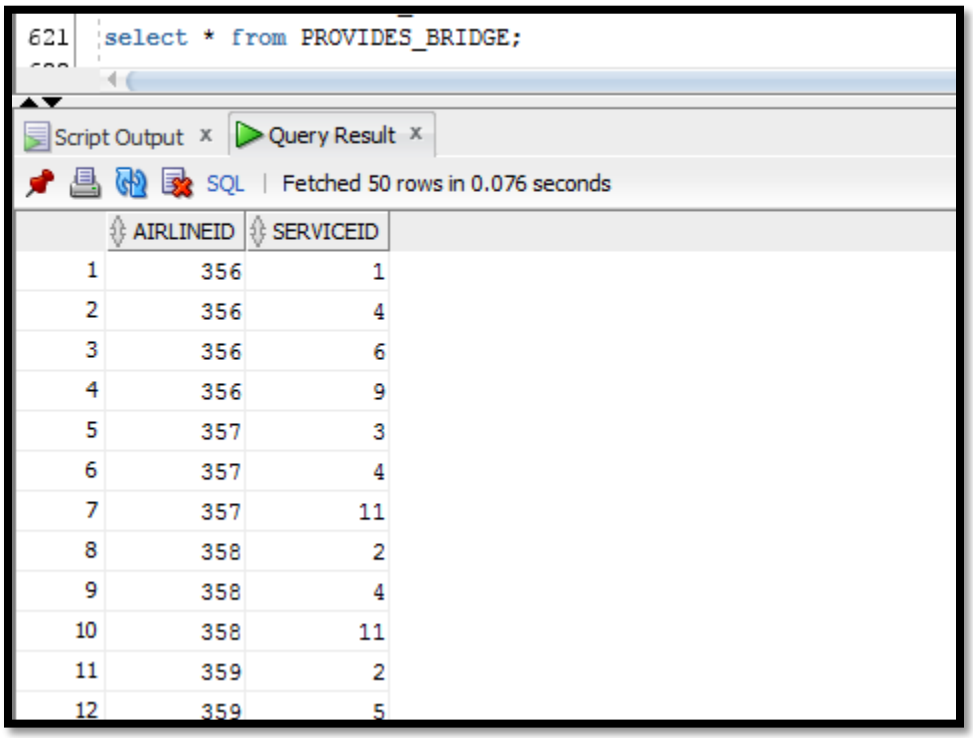
Screenshot showing airline_dim table

```
620  select * from AIRLINE_DIM;
621
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.047 seconds

	AIRLINEID	NAME	COUNTRY	ACTIVE	WEIGHTFACTOR	SERVICEGROUPLIST
1	1	Private flight	Unknown	Y	0.33	1_4_10
2	2	135 Airways	United States	N	0.25	1_4_7_8
3	3	1Time Airline	South Africa	Y	0.33	3_5_10
4	4	2 Sqn No 1 Elementary Flying Training School	United Kingdom	N	0.25	3_5_6_9
5	5	213 Flight Unit	Russia	N	0.25	3_4_6_9
6	6	223 Flight Unit State Airline	Russia	N	0.25	1_4_7_8
7	7	224th Flight Unit	Russia	N	0.25	2_5_6_8
8	8	247 Jet Ltd	United Kingdom	N	0.33	3_4_11
9	9	3D Aviation	United States	N	0.33	1_5_10
10	10	40-Mile Air	United States	Y	0.33	3_5_10
11	11	4D Air	Thailand	N	0.25	2_4_6_9
12	12	611897 Alberta Limited	Canada	N	0.33	1_4_11
13	13	Ansett Australia	Australia	Y	0.33	3_4_11

Screenshot showing provides_bridge table

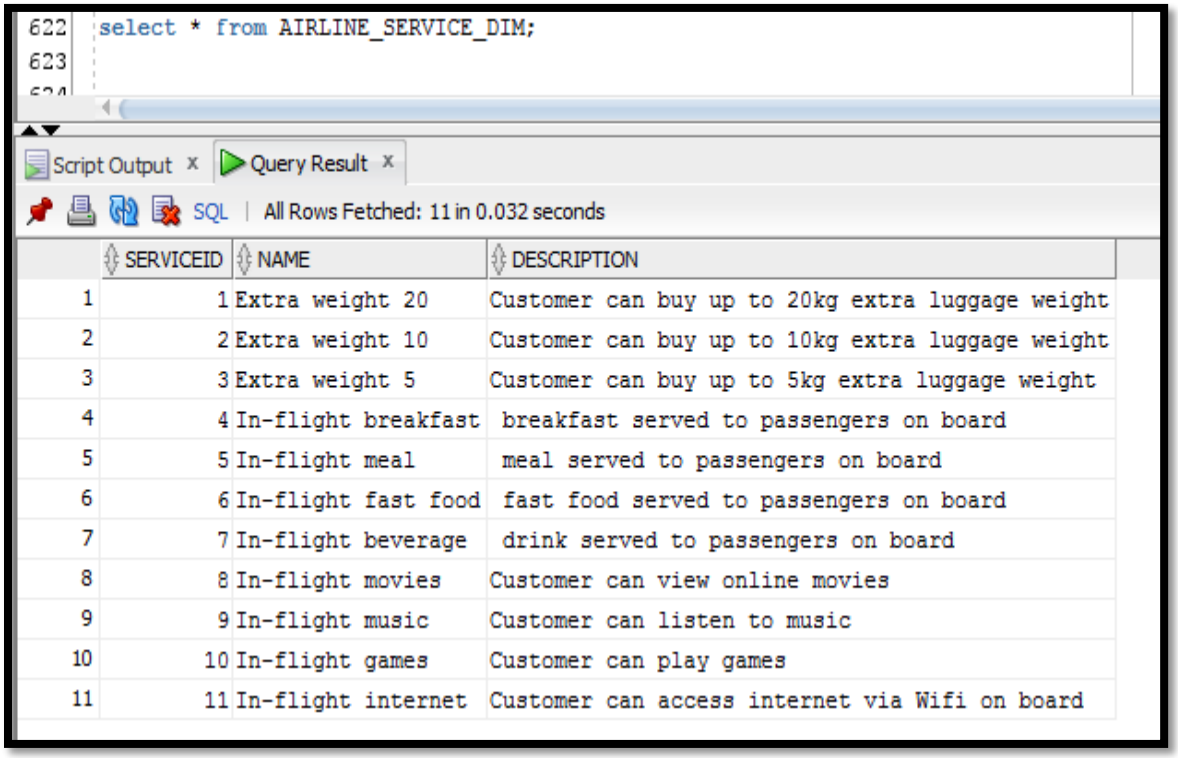


```
621  select * from PROVIDES_BRIDGE;
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.076 seconds

	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

Screenshot showing airline_service_dim table

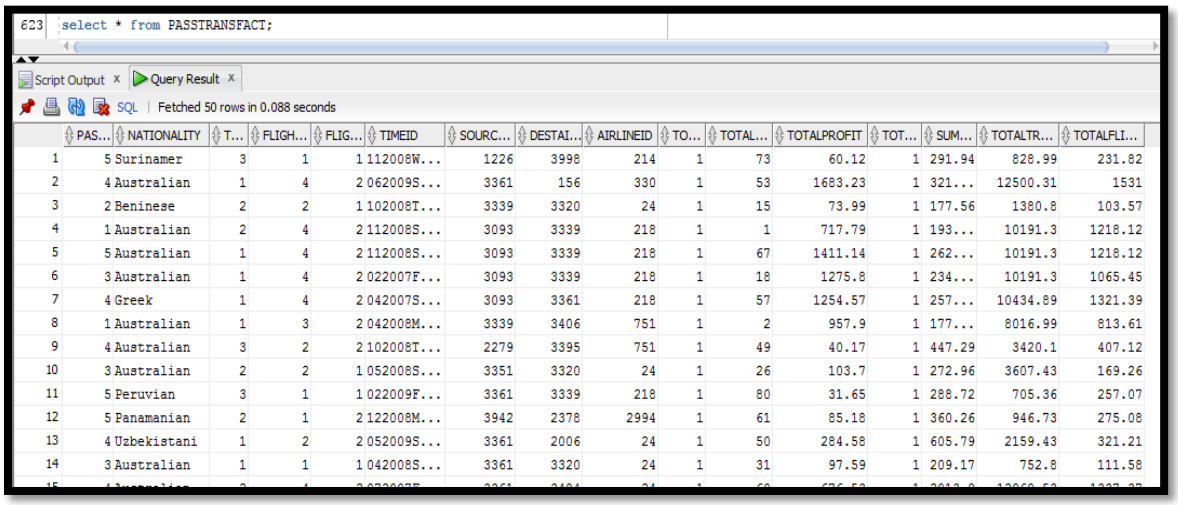


```
622  select * from AIRLINE_SERVICE_DIM;
623
```

Script Output | Query Result | SQL | All Rows Fetched: 11 in 0.032 seconds

	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

Screenshot showing PassTransFact table

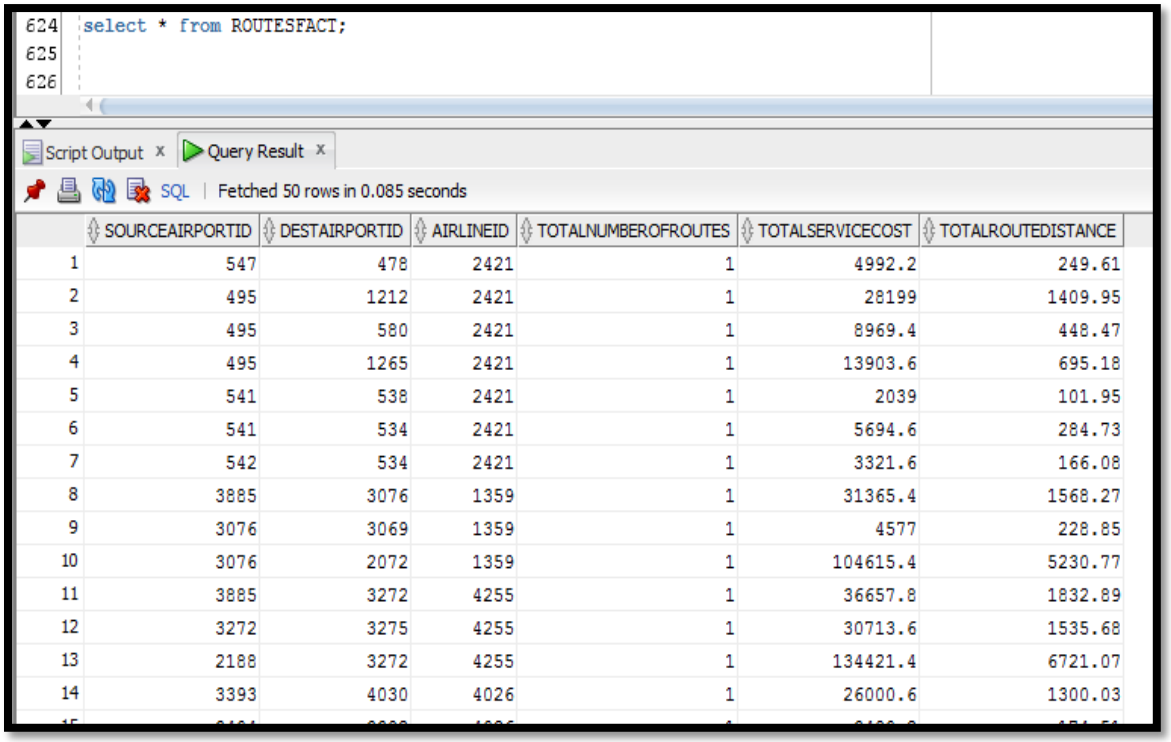


```
623  select * from PASSTRANSFACT;
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.088 seconds

	PAS...	NATIONALITY	T...	FLIGH...	FLIG...	TIMEID	SOURC...	DESTAI...	AIRLINEID	TO...	TOTAL...	TOTALPROFIT	TOT...	SUM...	TOTALTR...	TOTALFLI...
1	5	Surinamer	3	1	1	112008W...	1226	3998	214	1	73	60.12	1	291.94	828.99	231.82
2	4	Australian	1	4	2	062009S...	3361	156	330	1	53	1683.23	1	321...	12500.31	1531
3	2	Beninese	2	2	1	102008T...	3339	3320	24	1	15	73.99	1	177.56	1380.8	103.57
4	1	Australian	2	4	2	112008S...	3093	3339	218	1	1	717.79	1	193...	10191.3	1218.12
5	5	Australian	1	4	2	112008S...	3093	3339	218	1	67	1411.14	1	262...	10191.3	1218.12
6	3	Australian	1	4	2	022007F...	3093	3339	218	1	18	1275.8	1	234...	10191.3	1065.45
7	4	Greek	1	4	2	042007S...	3093	3361	218	1	57	1254.57	1	257...	10434.89	1321.39
8	1	Australian	1	3	2	042008M...	3339	3406	751	1	2	957.9	1	177...	8016.99	813.61
9	4	Australian	3	2	2	102008T...	2279	3395	751	1	49	40.17	1	447.29	3420.1	407.12
10	3	Australian	2	2	1	052008S...	3351	3320	24	1	26	103.7	1	272.96	3607.43	169.26
11	5	Peruvian	3	1	1	022009F...	3361	3339	218	1	80	31.65	1	288.72	705.36	257.07
12	5	Panamanian	2	1	2	122008M...	3942	2378	2994	1	61	85.18	1	360.26	946.73	275.08
13	4	Uzbekistani	1	2	2	052009S...	3361	2006	24	1	50	284.58	1	605.79	2159.43	321.21
14	3	Australian	1	1	1	042008S...	3361	3320	24	1	31	97.59	1	209.17	752.8	111.58

Screenshot showing RoutesFact table



```
624  select * from ROUTESFACT;
625
626
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.085 seconds

	SOURCEAIRPORTID	DESTAIRPORTID	AIRLINEID	TOTALNUMBEROFROUTES	TOTALSERVICECOST	TOTALROUTEDISTANCE
1	547	478	2421	1	4992.2	249.61
2	495	1212	2421	1	28199	1409.95
3	495	580	2421	1	8969.4	448.47
4	495	1265	2421	1	13903.6	695.18
5	541	538	2421	1	2039	101.95
6	541	534	2421	1	5694.6	284.73
7	542	534	2421	1	3321.6	166.08
8	3885	3076	1359	1	31365.4	1568.27
9	3076	3069	1359	1	4577	228.85
10	3076	2072	1359	1	104615.4	5230.77
11	3885	3272	4255	1	36657.8	1832.89
12	3272	3275	4255	1	30713.6	1535.68
13	2188	3272	4255	1	134421.4	6721.07
14	3393	4030	4026	1	26000.6	1300.03

2. Task C.2

2.1.Simple reports

2.1.1. Report 1: Show All Report

a) Query Question

What is the total number of passengers who departed from cities of Australia each year?

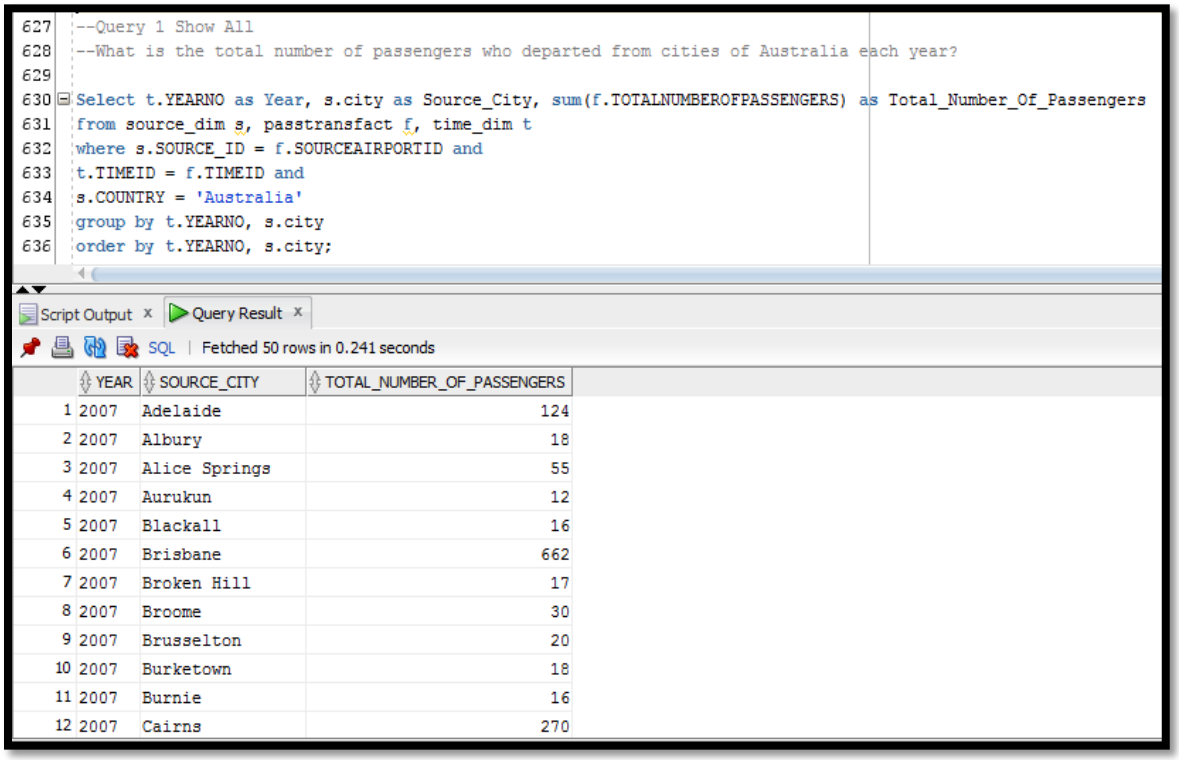
b) Explanation

The top management would be interested in the total number of passengers travelling in a particular year from a particular country to take decisions related to the frequency of the flights from the source airport.

c) SQL command

```
Select t.YEARNO as Year, s.city as Source_City,
sum(f.TOTALNUMBEROFPASSENGERS) as Total_Number_Of_Passengers
from source_dim s, passtransfact f, time_dim t
where s.SOURCE_ID = f.SOURCEAIRPORTID and
t.TIMEID = f.TIMEID and
s.COUNTRY = 'Australia'
group by t.YEARNO, s.city
order by t.YEARNO, s.city;
```

d) Screenshot of Query result



```
627  --Query 1 Show All
628  --What is the total number of passengers who departed from cities of Australia each year?
629
630  Select t.YEARNO as Year, s.city as Source_City, sum(f.TOTALNUMBEROFPASSENGERS) as Total_Number_Of_Passengers
631  from source_dim s, passtransfact f, time_dim t
632  where s.SOURCE_ID = f.SOURCEAIRPORTID and
633  t.TIMEID = f.TIMEID and
634  s.COUNTRY = 'Australia'
635  group by t.YEARNO, s.city
636  order by t.YEARNO, s.city;
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.241 seconds

	YEAR	SOURCE_CITY	TOTAL_NUMBER_OF_PASSENGERS
1	2007	Adelaide	124
2	2007	Albury	18
3	2007	Alice Springs	55
4	2007	Aurukun	12
5	2007	Blackall	16
6	2007	Brisbane	662
7	2007	Broken Hill	17
8	2007	Broome	30
9	2007	Brusselton	20
10	2007	Burketown	18
11	2007	Burnie	16
12	2007	Cairns	270

2.1.2. Report 2: Top k Report

a) Query Question

Which are the top 3 airlines with maximum profit for Australia in the year 2007 and show whether they are active or not?

b) Explanation

The top management would be interested in determing which of the active airlines is generating maximum profit in a particular year and particular country.

c) SQL command

```
SELECT *
FROM
(SELECT a.Name, a.ACTIVE, SUM(f.TOTALPROFIT) AS Profit,
RANK() OVER (ORDER BY SUM(f.TOTALPROFIT) DESC ) AS
Airline_RANK
 FROM passtransfact f, AIRLINE_DIM a, TIME_DIM t
 WHERE f.TIMEID = t.TIMEID
AND f.AIRLINEID = a.AIRLINEID
AND a.Country = 'Australia'
AND t.YEARNO = 2007
GROUP BY a.Name, a.ACTIVE)
WHERE Airline_RANK <= 3;
```

d) Screenshot of Query result

```
641 SELECT *
642 FROM
643 (SELECT a.Name, a.ACTIVE, SUM(f.TOTALPROFIT) AS Profit,
644 RANK() OVER (ORDER BY SUM(f.TOTALPROFIT) DESC ) AS
645 Airline_RANK
646  FROM passtransfact f, AIRLINE_DIM a, TIME_DIM t
647  WHERE f.TIMEID = t.TIMEID
648 AND f.AIRLINEID = a.AIRLINEID
649 AND a.Country = 'Australia'
650 AND t.YEARNO = 2007
651 GROUP BY a.Name, a.ACTIVE)
652 WHERE Airline_RANK <= 3;
653
```

Script Output x | Query Result x

SQL | Fetched 50 rows in 0.241 seconds

	YEAR	SOURCE_CITY	TOTAL_NUMBER_OF_PASSENGERS
1	2007	Adelaide	124
2	2007	Albury	18
3	2007	Alice Springs	55
4	2007	Aurukun	12
5	2007	Blackall	16
6	2007	Brisbane	662
7	2007	Broken Hill	17
8	2007	Broome	30
9	2007	Brusselton	20
10	2007	Burketown	18
11	2007	Burnie	16

2.1.3. Report 3: Top n% Report

a) Query Question

Calculate the top 5% of the total service cost of qantas airlines from different source cities?

b) Explanation

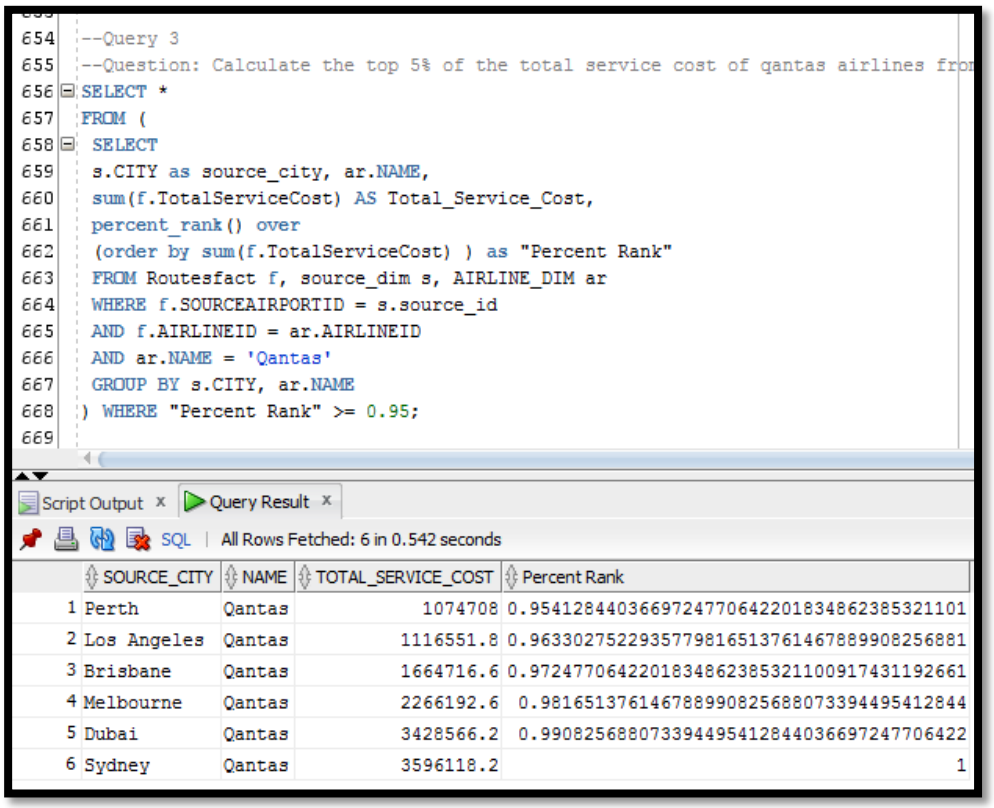
The top management would be interested in knowing the top 5% airlines based on their spending on providing services.

c) SQL command

```
SELECT *
FROM (
SELECT
s.CITY as source_city, ar.NAME,
```

```
sum(f.TotalServiceCost) AS Total_Service_Cost,
percent_rank() over
(order by sum(f.TotalServiceCost) ) as "Percent Rank"
FROM Routesfact f, source_dim s, AIRLINE_DIM ar
WHERE f.SOURCEAIRPORTID = s.source_id
AND f.AIRLINEID = ar.AIRLINEID
AND ar.NAME = 'Qantas'
GROUP BY s.CITY, ar.NAME
) WHERE "Percent Rank" >= 0.95;
```

d) Screenshot of Query result



```
654  --Query 3
655  --Question: Calculate the top 5% of the total service cost of qantas airlines fro
656  SELECT *
657  FROM (
658   SELECT
659   s.CITY as source_city, ar.NAME,
660   sum(f.TotalServiceCost) AS Total_Service_Cost,
661   percent_rank() over
662   (order by sum(f.TotalServiceCost) ) as "Percent Rank"
663   FROM Routesfact f, source_dim s, AIRLINE_DIM ar
664   WHERE f.SOURCEAIRPORTID = s.source_id
665   AND f.AIRLINEID = ar.AIRLINEID
666   AND ar.NAME = 'Qantas'
667   GROUP BY s.CITY, ar.NAME
668  ) WHERE "Percent Rank" >= 0.95;
669
```

Script Output | Query Result | SQL | All Rows Fetched: 6 in 0.542 seconds

	SOURCE_CITY	NAME	TOTAL_SERVICE_COST	Percent Rank
1	Perth	Qantas	1074708	0.954128440366972477064220183486238532110
2	Los Angeles	Qantas	1116551.8	0.963302752293577981651376146788990825688
3	Brisbane	Qantas	1664716.6	0.972477064220183486238532110091743119266
4	Melbourne	Qantas	2266192.6	0.981651376146788990825688073394495412844
5	Dubai	Qantas	3428566.2	0.990825688073394495412844036697247706422
6	Sydney	Qantas	3596118.2	1

2.2.More Advanced Reports

2.2.1. Report 4: City-to-City Routes' Report

a) Query question

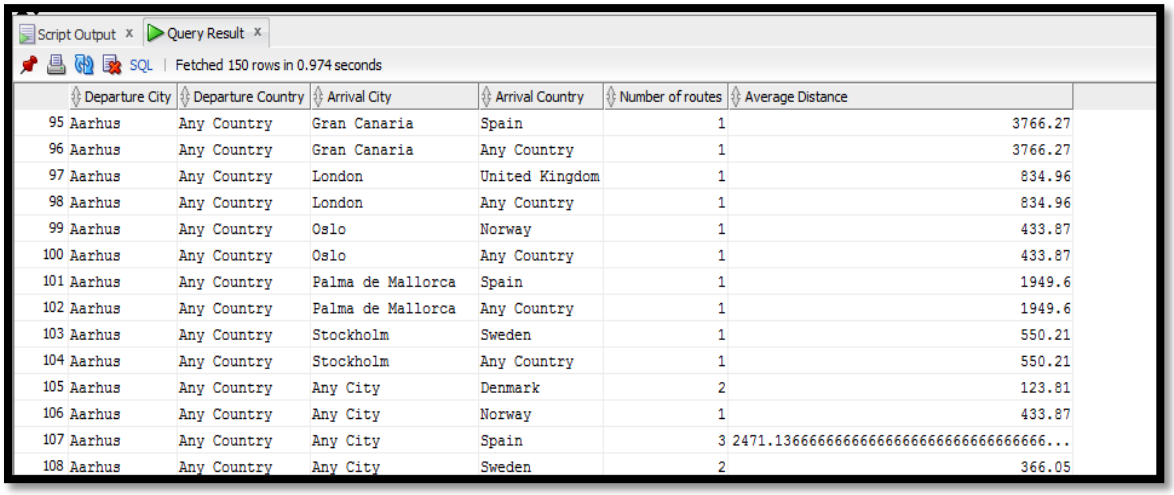
What is the average route distance between source cities and destination cities?

b) SQL command

```
select decode(grouping(s.CITY),1,'Any City',s.CITY) as "Departure City",
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) as "Departure Country",
decode(grouping(d.CITY),1,'Any City',d.CITY) as "Arrival City",
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) as "Arrival Country",
```

```
sum(r.TOTALNUMBEROFROUTES) as "Number of routes",
sum(r.TOTALROUTEDISTANCE)/sum(r.TOTALNUMBEROFROUTES) as "Average
Distance"
from destination_dim d, source_dim s, routesfact r
where d.DESTINATION_ID= r.DESTAIRPORTID
and s.SOURCE_ID = r.SOURCEAIRPORTID
group by cube(s.CITY,s.COUNTRY,d.CITY, d.COUNTRY)
order by s.CITY,s.COUNTRY,d.CITY,d.COUNTRY;
```

c) Screenshot of query result



Script Output | Query Result | SQL | Fetched 150 rows in 0.974 seconds

	Departure City	Departure Country	Arrival City	Arrival Country	Number of routes	Average Distance
95	Aarhus	Any Country	Gran Canaria	Spain	1	3766.27
96	Aarhus	Any Country	Gran Canaria	Any Country	1	3766.27
97	Aarhus	Any Country	London	United Kingdom	1	834.96
98	Aarhus	Any Country	London	Any Country	1	834.96
99	Aarhus	Any Country	Oslo	Norway	1	433.87
100	Aarhus	Any Country	Oslo	Any Country	1	433.87
101	Aarhus	Any Country	Palma de Mallorca	Spain	1	1949.6
102	Aarhus	Any Country	Palma de Mallorca	Any Country	1	1949.6
103	Aarhus	Any Country	Stockholm	Sweden	1	550.21
104	Aarhus	Any Country	Stockholm	Any Country	1	550.21
105	Aarhus	Any Country	Any City	Denmark	2	123.81
106	Aarhus	Any Country	Any City	Norway	1	433.87
107	Aarhus	Any Country	Any City	Spain	3	2471.13666666666666666666666666666666...
108	Aarhus	Any Country	Any City	Sweden	2	366.05

2.2.2. Report 5: Airline's Report

a) Query question

What is the average agent profit every year for each airline from source countries to destination countries with respect to each flight type?

b) SQL command

```
SELECT t.YEARNO, ar.NAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
sum(f.TOTALPROFIT) AS "Average Agent Profit"
FROM passtransfact f, time_dim t, airline_dim ar, Flight_Type_dim ft, source_dim s,
destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.AIRLINEID = ar.AIRLINEID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
```

```
GROUP by  t.YEARNO, ar.NAME, rollup(ft.FLIGHTTYPEDESC, s.COUNTRY,
d.COUNTRY)
order by t.YEARNO, ar.NAME, ft.FLIGHTTYPEDESC, s.COUNTRY, d.COUNTRY;
```

c) Screenshot of query result

Script Output | Query Result | SQL | Fetched 50 rows in 0.206 seconds

	YEARNO	NAME	FLIGHTTYPE	SOURCECOUNTRY	DESTINATIONCOUNTRY	Number of Transactions	Average Agent Profit
1	2007	Aegean Airlines	International	Greece	Russia	1	627.03
2	2007	Aegean Airlines	International	Greece	Any Country	1	627.03
3	2007	Aegean Airlines	International	Any Country	Any Country	1	627.03
4	2007	Aegean Airlines	All Flight Type	Any Country	Any Country	1	627.03
5	2007	Aer Lingus	International	Ireland	United Kingdom	1	50.39
6	2007	Aer Lingus	International	Ireland	Any Country	1	50.39
7	2007	Aer Lingus	International	Any Country	Any Country	1	50.39
8	2007	Aer Lingus	All Flight Type	Any Country	Any Country	1	50.39
9	2007	Aero Condor Peru	Domestic	Australia	Australia	203	37175.32
10	2007	Aero Condor Peru	Domestic	Australia	Any Country	203	37175.32
11	2007	Aero Condor Peru	Domestic	Any Country	Any Country	203	37175.32
12	2007	Aero Condor Peru	All Flight Type	Any Country	Any Country	203	37175.32
13	2007	AeroMéxico	Domestic	Mexico	Mexico	2	185.12
14	2007	AeroMéxico	Domestic	Mexico	Any Country	2	185.12

2.2.3. Report 6: Flight's Report

a) Query question

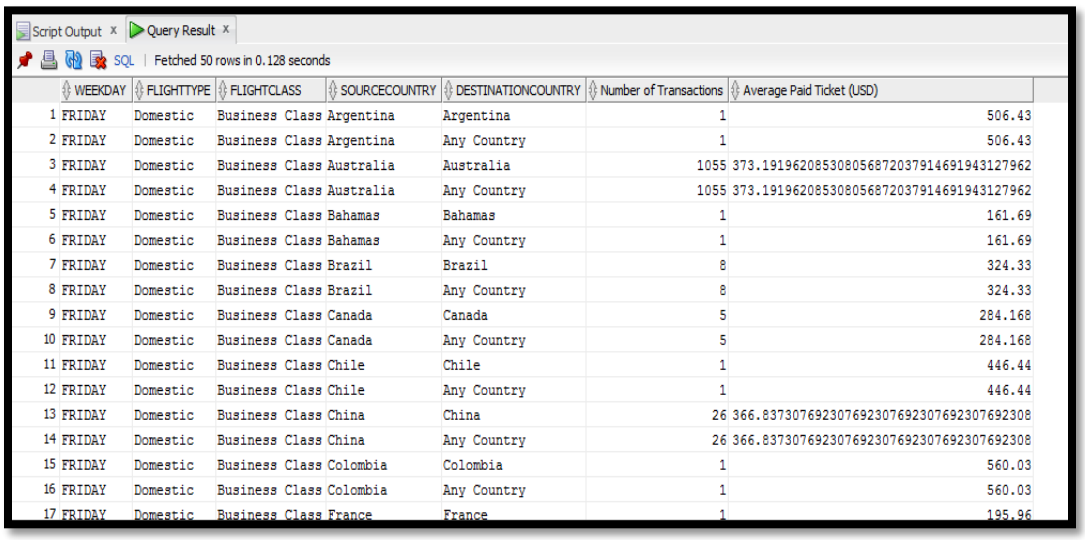
What is the average paid ticket per weekday for each flight type and class from source country to destination country?

b) SQL command

```
SELECT t.WEEKDAY,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
(sum(f.SUMOFTOTALPAID)/ sum(f.TOTALNUMBEROFTRANSACTIONS)) AS
"Average Paid Ticket (USD)"
 FROM passtransfact f, time_dim t, TRAVEL_CLASS_DIM tr, Flight_Type_dim ft,
 source_dim s, destination_dim d
 WHERE f.TIMEID = t.TIMEID
 and f.TRAVELID = tr.TRAVELID
 and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
 and f.SOURCEAIRPORTID = s.SOURCE_ID
 and f.DESTAIRPORTID = d.DESTINATION_ID
```

```
GROUP by  t.WEEKDAY,  rollup(ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC,
s.COUNTRY, d.COUNTRY)
order by t.WEEKDAY, ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC, s.COUNTRY,
d.COUNTRY;
```

c) Screenshot of query result



Script Output × Query Result ×

SQL | Fetched 50 rows in 0.128 seconds

	WEEKDAY	FLIGHTTYPE	FLIGHTCLASS	SOURCECOUNTRY	DESTINATIONCOUNTRY	Number of Transactions	Average Paid Ticket (USD)
1	FRIDAY	Domestic	Business Class	Argentina	Argentina	1	506.43
2	FRIDAY	Domestic	Business Class	Argentina	Any Country	1	506.43
3	FRIDAY	Domestic	Business Class	Australia	Australia	1055	373.191962085308056872037914691943127962
4	FRIDAY	Domestic	Business Class	Australia	Any Country	1055	373.191962085308056872037914691943127962
5	FRIDAY	Domestic	Business Class	Bahamas	Bahamas	1	161.69
6	FRIDAY	Domestic	Business Class	Bahamas	Any Country	1	161.69
7	FRIDAY	Domestic	Business Class	Brazil	Brazil	8	324.33
8	FRIDAY	Domestic	Business Class	Brazil	Any Country	8	324.33
9	FRIDAY	Domestic	Business Class	Canada	Canada	5	284.168
10	FRIDAY	Domestic	Business Class	Canada	Any Country	5	284.168
11	FRIDAY	Domestic	Business Class	Chile	Chile	1	446.44
12	FRIDAY	Domestic	Business Class	Chile	Any Country	1	446.44
13	FRIDAY	Domestic	Business Class	China	China	26	366.837307692307692307692307692307692308
14	FRIDAY	Domestic	Business Class	China	Any Country	26	366.837307692307692307692307692307692308
15	FRIDAY	Domestic	Business Class	Colombia	Colombia	1	560.03
16	FRIDAY	Domestic	Business Class	Colombia	Any Country	1	560.03
17	FRIDAY	Domestic	Business Class	France	France	1	195.96

2.3.Reports with Cube and Rollup and the comparison

2.3.1. Report 7: Report using Cube

a) Query question

What is the total revenue generated in the month of September for each flight type per travel class?

b) Explanation

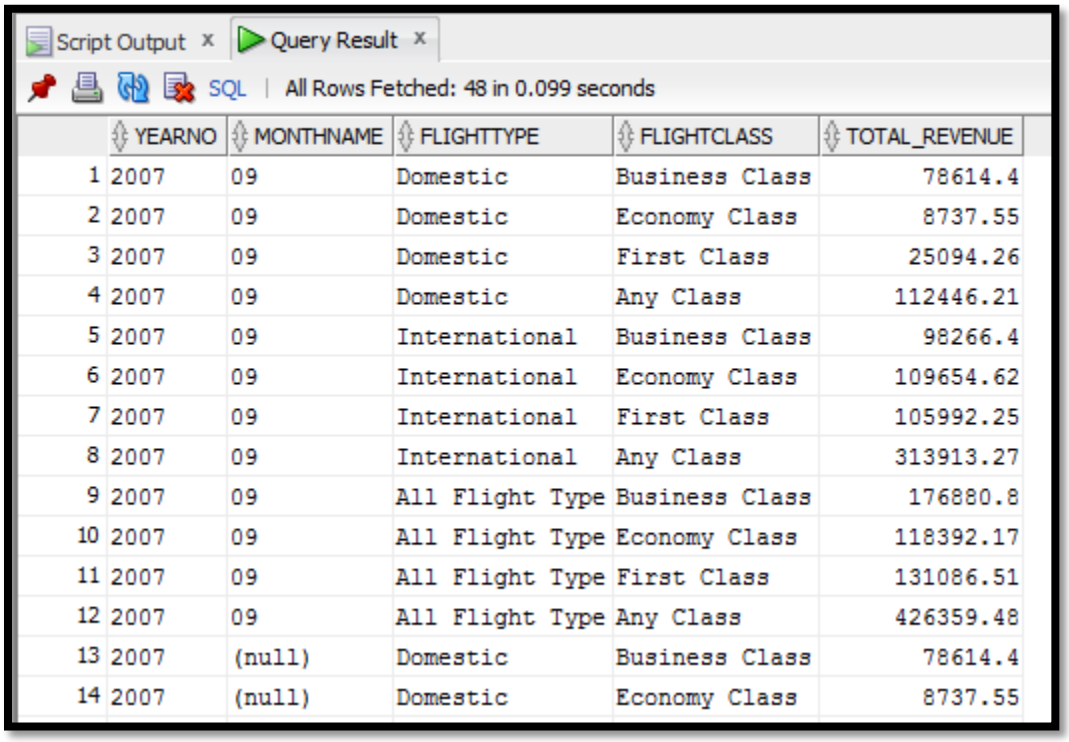
The top management would be interested in finding the total revenue generated in a particular month, and a particular year and produced for all possible combinations of subtotals for the attributes Year, Monthname, flightTypeDesc and ClassTypeDesc

c) SQL command

```
SELECT t.YEARNO, t.MONTHNAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
sum(f.SUMOFTOTALPAID) as Total_Revenue
FROM TIME_DIM t, FLIGHT_TYPE_DIM ft, TRAVEL_CLASS_DIM tr,
PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
```

```
AND tr.TRAVELID = f.TRAVELID
and t.YEARNO = 2007
and t.MONTHNAME = 09
GROUP BY CUBE(t.YEARNO, t.MONTHNAME, ft.FLIGHTTYPEDESC,
tr.CLASSTYPEDESC)
Order by t.YEARNO, t.MONTHNAME;
```

d) Screenshot of query result



Script Output | Query Result | SQL | All Rows Fetched: 48 in 0.099 seconds

	YEARNO	MONTHNAME	FLIGHTTYPE	FLIGHTCLASS	TOTAL_REVENUE
1	2007	09	Domestic	Business Class	78614.4
2	2007	09	Domestic	Economy Class	8737.55
3	2007	09	Domestic	First Class	25094.26
4	2007	09	Domestic	Any Class	112446.21
5	2007	09	International	Business Class	98266.4
6	2007	09	International	Economy Class	109654.62
7	2007	09	International	First Class	105992.25
8	2007	09	International	Any Class	313913.27
9	2007	09	All Flight Type	Business Class	176880.8
10	2007	09	All Flight Type	Economy Class	118392.17
11	2007	09	All Flight Type	First Class	131086.51
12	2007	09	All Flight Type	Any Class	426359.48
13	2007	(null)	Domestic	Business Class	78614.4
14	2007	(null)	Domestic	Economy Class	8737.55

2.3.2. Report 8: Report using Rollup

a) Query question

What is the total travel distance travelled by selected airlines ('China Eastern Airlines', 'IndiGo Airlines', 'Virgin America') for each year per flight type?

b) Explanation

The top management is interested in determining the total travel distance of selected few airlines for particular year for domestic and international flights.

c) SQL command

```
SELECT
decode(grouping(t.YEARNO),1,'All Years',t.YEARNO) as Year,
decode(grouping(a.NAME),1,'All AirLines',a.NAME) AS AirlineName,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Types',ft.FLIGHTTYPEDESC) AS
FlightType,
```

```
sum(f.TotalTravelDistance) as Total_Distance
FROM TIME_DIM t, AIRLINE_DIM a, FLIGHT_TYPE_DIM ft, PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND a.airlineid = f.airlineid
and a.name IN ('China Eastern Airlines','IndiGo Airlines','Virgin America')
GROUP BY ROLLUP(t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC)
Order by t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC;
```

d) Screenshot of query result

Script Output | Query Result

SQL | All Rows Fetched: 25 in 0.041 seconds

	YEAR	AIRLINENAME	FLIGHTTYPE	TOTAL_DISTANCE
1	2007	China Eastern Airlines	Domestic	3783.2
2	2007	China Eastern Airlines	International	842980.53
3	2007	China Eastern Airlines	All Types	846763.73
4	2007	IndiGo Airlines	Domestic	3175.8
5	2007	IndiGo Airlines	All Types	3175.8
6	2007	Virgin America	Domestic	3883.2
7	2007	Virgin America	All Types	3883.2
8	2007	All AirLines	All Types	853822.73
9	2008	China Eastern Airlines	Domestic	10559.2
10	2008	China Eastern Airlines	International	269343.67
11	2008	China Eastern Airlines	All Types	279902.87
12	2008	IndiGo Airlines	Domestic	3822.8
13	2008	IndiGo Airlines	All Types	3822.8
14	2008	Virgin America	Domestic	7143.01
15	2008	Virgin America	All Types	7143.01
16	2008	All AirLines	All Types	290868.68
17	2009	China Eastern Airlines	Domestic	12006.96

e) Comparison between report 7 and report 8

- Cube: Cube generates result set that shows aggregate for all possible combination of values in selected column. In the event, that we had used Cube in Report 8, additional rows showing all possible combinations and subtotals of specified attributes year, Airline name, flightTypeDesc

- RollUp: RollUp generates result set that shows aggregate hierarchy of values in the selected column. In the event, that we had used Rollup in Report 7, the rows showing subtotals and grand totals about Year, Monthname, flightTypeDesc and ClassTypeDesc attributes will not be present. Rollup is a summarized report as compared to Cube.

2.4.Reports with Moving and Cumulative Aggregates

2.4.1. Report 9

a) Query question

What are the total and cumulative monthly profits of small flight distance departing from Sydney airport in 2007?

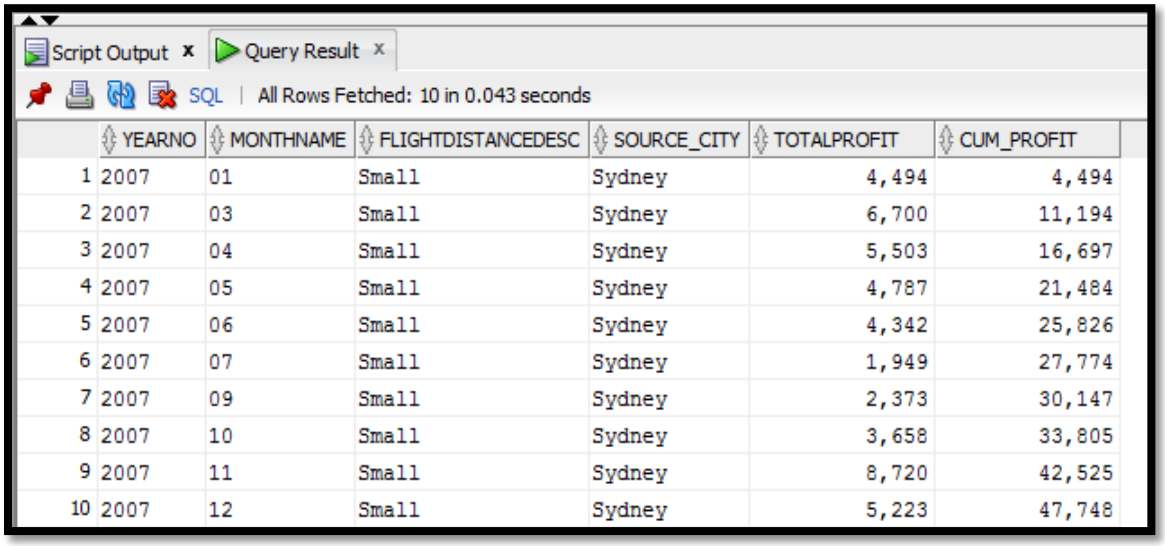
b) Explanation

The top management would be interested in determining the total and cumulative monthly profits for flights covering small distance departing from a particular source country in particular year.

c) SQL Command

```
SELECT t.yearNo, t.monthname, f.flightdistancedesc, s.city as source_city,
TO_CHAR (SUM(p.TotalProfit), '9,999,999,999') AS TotalProfit,
TO_CHAR (SUM(SUM(p.TotalProfit)) OVER
(ORDER BY t.yearNo, t.monthname, f.flightdistancedesc, s.city ROWS UNBOUNDED
PRECEDING),
'9,999,999,999') AS CUM_profit
FROM time_dim t, passtransfact p, flight_distance_dim f, source_dim s
WHERE t.TIMEID = p.TIMEID
AND s.SOURCE_ID = p.SOURCEAIRPORTID
and f.FLIGHTDISTANCEID = p.FLIGHTDISTANCEID
AND t.YEARNO = 2007
AND f.FLIGHTDISTANCEDESC = 'Small'
and s.CITY = 'Sydney'
GROUP BY t.yearNo, t.monthname, f.flightdistancedesc, s.city;
```

d) Screenshot of query result



Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.043 seconds

	YEARNO	MONTHNAME	FLIGHTDISTANCEDESC	SOURCE_CITY	TOTALPROFIT	CUM_PROFIT
1	2007	01	Small	Sydney	4,494	4,494
2	2007	03	Small	Sydney	6,700	11,194
3	2007	04	Small	Sydney	5,503	16,697
4	2007	05	Small	Sydney	4,787	21,484
5	2007	06	Small	Sydney	4,342	25,826
6	2007	07	Small	Sydney	1,949	27,774
7	2007	09	Small	Sydney	2,373	30,147
8	2007	10	Small	Sydney	3,658	33,805
9	2007	11	Small	Sydney	8,720	42,525
10	2007	12	Small	Sydney	5,223	47,748

2.4.2. Report 10

a) Query question

What are the total and moving 3 monthly transactions of Australian passengers in 2009?

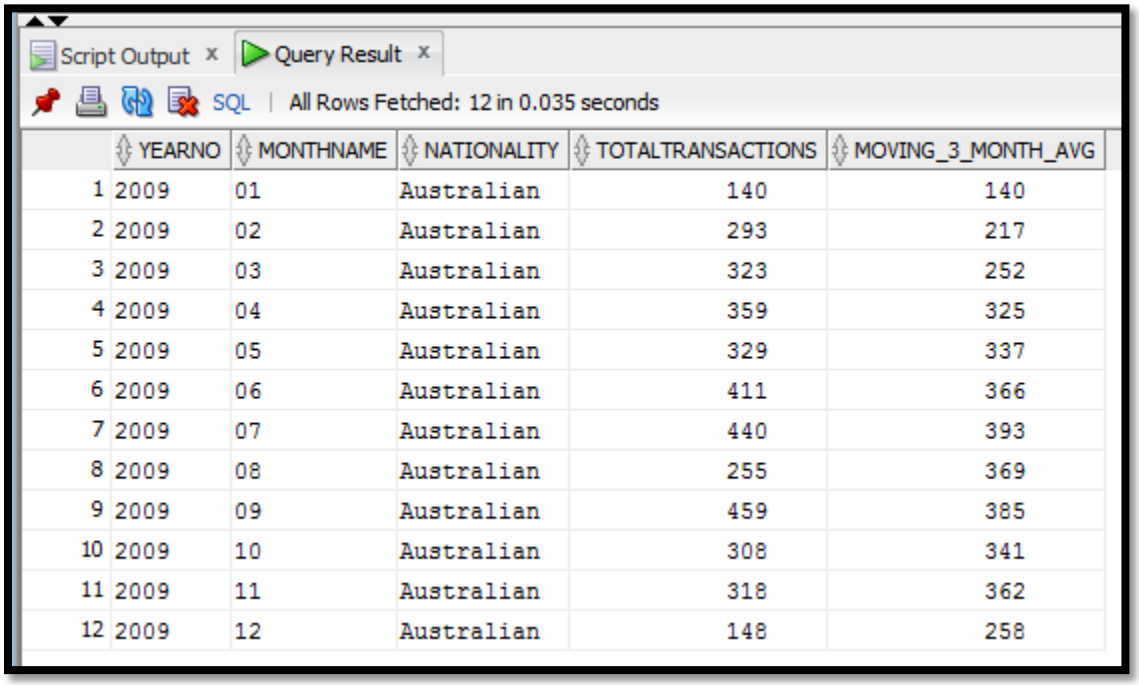
b) Explanation

The top management would be interested in the determining the total and moving 3 monthly transactions from passengers of a particular nationality in a particular year.

c) SQL Command

```
SELECT t.YEARNO,t.MONTHNAME, n.NATIONALITY,
TO_CHAR (SUM(p.TotalNumberofTransactions), '9,999,999,999') AS TotalTransactions,
TO_CHAR (AVG(SUM(p.TotalNumberofTransactions)) OVER
(ORDER BY t.YEARNO,t.MONTHNAME, n.NATIONALITY
rows 2 preceding),
 '9,999,999,999') AS moving_3_month_avg
FROM TIME_DIM t, PASSTRANSFACT p, NATIONALITY_DIM n
WHERE t.TIMEID = p.TIMEID
AND n.NATIONALITY = p.NATIONALITY
AND t.YEARNO=2009
and n.NATIONALITY= 'Australian'
GROUP BY t.YEARNO,t.MONTHNAME, n.NATIONALITY;
```

d) Screenshot of query result



Script Output | Query Result

SQL | All Rows Fetched: 12 in 0.035 seconds

	YEARNO	MONTHNAME	NATIONALITY	TOTALTRANSACTIONS	MOVING_3_MONTH_AVG
1	2009	01	Australian	140	140
2	2009	02	Australian	293	217
3	2009	03	Australian	323	252
4	2009	04	Australian	359	325
5	2009	05	Australian	329	337
6	2009	06	Australian	411	366
7	2009	07	Australian	440	393
8	2009	08	Australian	255	369
9	2009	09	Australian	459	385
10	2009	10	Australian	308	341
11	2009	11	Australian	318	362
12	2009	12	Australian	148	258

2.4.3. Report 11

a) Query question

What are the total and cumulative monthly profits from Middle Adult passengers travelling by Southwest Airlines every year?

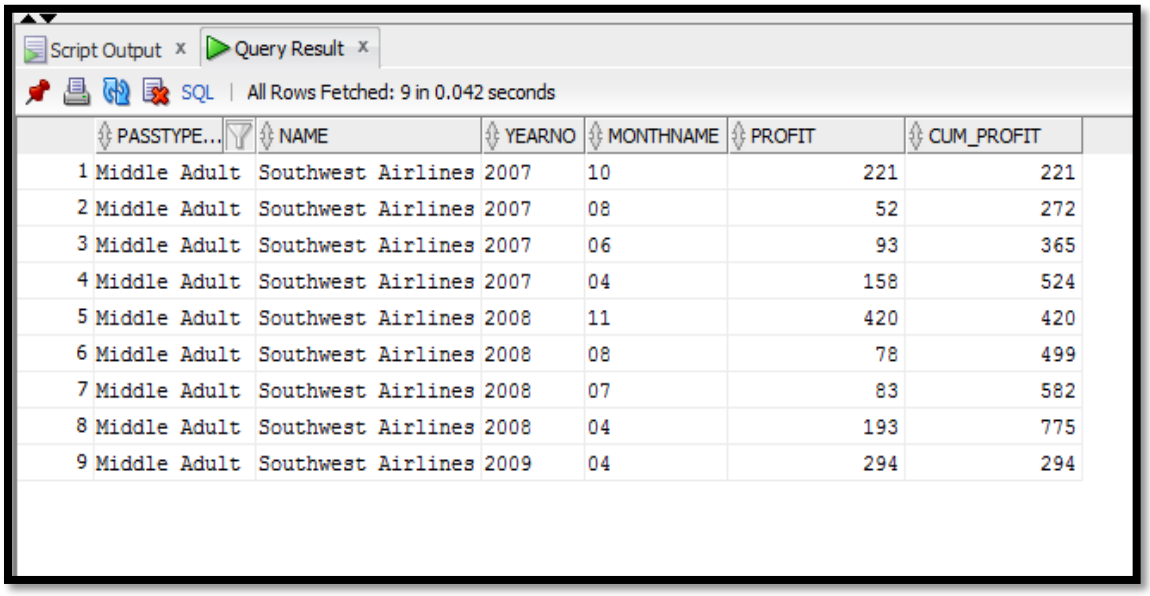
b) Explanation

The top management would be interested in determining the total and cumulative monthly profits generated by the particular airlines from passengers of particular age group every year.

c) SQL Command

```
SELECT p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME,
TO_CHAR (SUM(f.TOTALPROFIT), '9,999,999,999') AS PROFIT,
TO_CHAR (SUM(SUM(f.TOTALPROFIT)) OVER
(PARTITION BY t.YEARNO ORDER BY p.PASSTYPEDESC, a.NAME, t.YEARNO,
t.MONTHNAME desc
 ROWS UNBOUNDED PRECEDING),
'9,999,999,999') AS CUM_PROFIT
FROM PASSENGER_TYPE_DIM p, PASSTRANSFACT f, TIME_DIM t, AIRLINE_DIM
a
WHERE p.passtypeid = f.passtypeid
AND t.TIMEID = f.TIMEID
AND a.AIRLINEID = f.AIRLINEID
AND a.NAME = 'Southwest Airlines'
AND p.PASSTYPEDESC = 'Middle Adult'
GROUP BY p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME;
```

d) Screenshot of query result



Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.042 seconds

	PASSTYPE...	NAME	YEARNO	MONTHNAME	PROFIT	CUM_PROFIT
1	Middle Adult	Southwest Airlines	2007	10	221	221
2	Middle Adult	Southwest Airlines	2007	08	52	272
3	Middle Adult	Southwest Airlines	2007	06	93	365
4	Middle Adult	Southwest Airlines	2007	04	158	524
5	Middle Adult	Southwest Airlines	2008	11	420	420
6	Middle Adult	Southwest Airlines	2008	08	78	499
7	Middle Adult	Southwest Airlines	2008	07	83	582
8	Middle Adult	Southwest Airlines	2008	04	193	775
9	Middle Adult	Southwest Airlines	2009	04	294	294

2.4.4. Report 12

a) Query question

What are the total and moving 3 monthly number of passengers travelling in business class in flights departing from Canada?

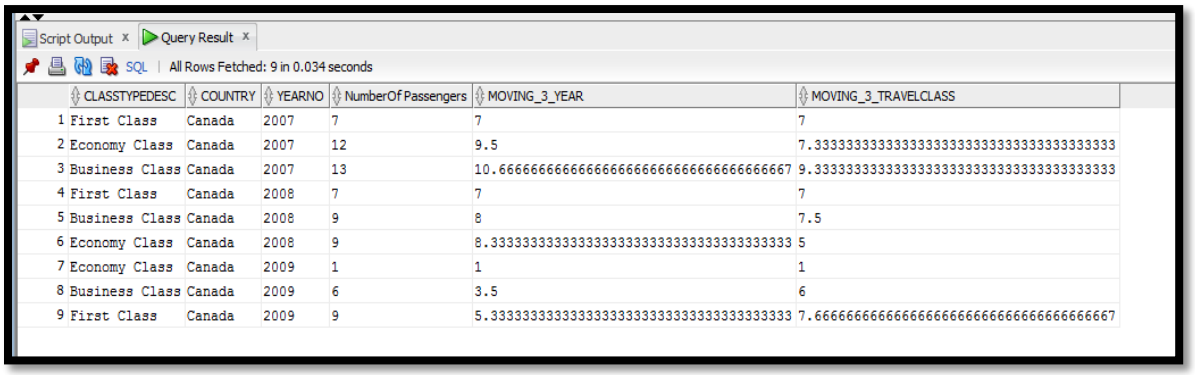
b) Explanation

The report would interest top management as it might want to determine the total number of passengers travelling in particular flight class who are departing from particular country on moving 3 monthly basis.

c) SQL Command

```
SELECT tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO,
TO_CHAR(SUM(f.TOTALNUMBEROFPASSENGERS)) AS "NumberOf Passengers",
TO_CHAR(AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(PARTITION BY t.YEARNO ORDER BY SUM(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_YEAR,
TO_CHAR (AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(partition by tr.classtypedesc ORDER BY sum(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_TravelClass
FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, SOURCE_DIM s, TIME_DIM t
WHERE t.TIMEID = f.TIMEID
AND tr.travelid = f.travelid
and s.source_id = f.sourceairportid
AND s.country = 'Canada'
GROUP BY tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO;
```

d) Screenshot of query result



Script Output | Query Result | SQL | All Rows Fetched: 9 in 0.034 seconds

	CLASSTYPEDESC	COUNTRY	YEARNO	NumberOf Passengers	MOVING_3_YEAR	MOVING_3_TRAVELCLASS
1	First Class	Canada	2007	7	7	7
2	Economy Class	Canada	2007	12	9.5	7.333333333333333333333333333333333333
3	Business Class	Canada	2007	13	10.666666666666666666666666666666666667	9.333333333333333333333333333333333333
4	First Class	Canada	2008	7	7	7
5	Business Class	Canada	2008	9	8	7.5
6	Economy Class	Canada	2008	9	8.333333333333333333333333333333333333	5
7	Economy Class	Canada	2009	1	1	1
8	Business Class	Canada	2009	6	3.5	6
9	First Class	Canada	2009	9	5.333333333333333333333333333333333333	7.666666666666666666666666666666666667

2.5.Reports with Rank and Percent Rank and the comparison

2.5.1. Report 13

a) Query question

What are the city ranks by total service cost of source airports in each country?

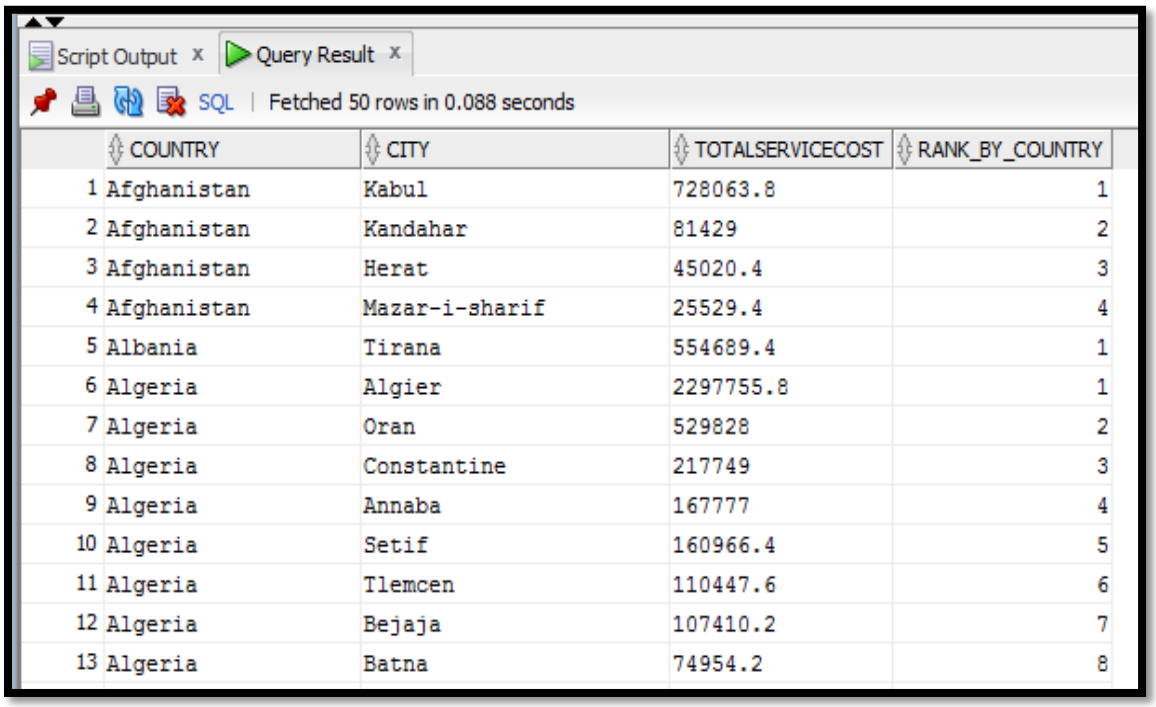
b) Explanation

The top management would be interested in finding the rank of cities in each country based on total service cost.

c) SQL Command

```
SELECT s.country,s.city,
TO_CHAR(SUM(TotalServiceCost)) AS TotalServiceCost,
RANK() OVER (PARTITION BY s.country
ORDER BY SUM(TotalServiceCost) DESC) AS RANK_BY_COUNTRY
FROM source_dim s, routesfact r
WHERE s.source_id=r.sourceairportid
GROUP BY s.country,s.city;
```

d) Screenshot of query result



Script Output | Query Result | SQL | Fetched 50 rows in 0.088 seconds

	COUNTRY	CITY	TOTALSERVICECOST	RANK_BY_COUNTRY
1	Afghanistan	Kabul	728063.8	1
2	Afghanistan	Kandahar	81429	2
3	Afghanistan	Herat	45020.4	3
4	Afghanistan	Mazar-i-sharif	25529.4	4
5	Albania	Tirana	554689.4	1
6	Algeria	Algier	2297755.8	1
7	Algeria	Oran	529828	2
8	Algeria	Constantine	217749	3
9	Algeria	Annaba	167777	4
10	Algeria	Setif	160966.4	5
11	Algeria	Tlemcen	110447.6	6
12	Algeria	Bejaja	107410.2	7
13	Algeria	Batna	74954.2	8

2.5.1. Report 14

a) Query question

What is the top 10% total revenue by nationality ('Angolan', 'Australian', 'British', 'Bangladeshi', 'Chinese', 'Batswana') and travel class?

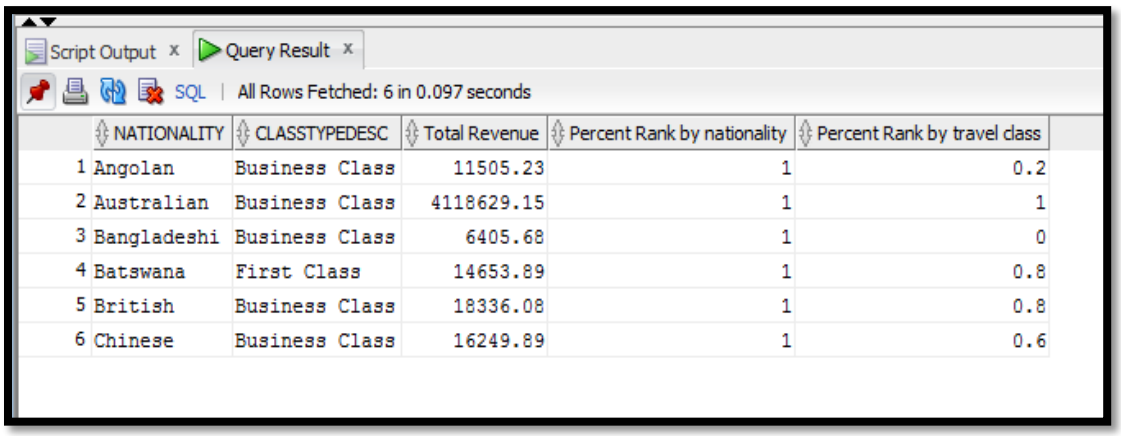
b) Explanation

The top management would be interested in finding the total revenue from passengers of different nationalities per travel class.

c) SQL Command

```
select *
from (
 SELECT
n.NATIONALITY, tr.classtypedesc,
 sum(f.SUMOFTOTALPAID) as "Total Revenue",
  percent_rank() over
 (partition by n.nationality order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by
nationality",
 percent_rank() over
 (partition by tr.classtypedesc order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by
travel class"
 FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, NATIONALITY_DIM n
 WHERE tr.TRAVELID = f.TRAVELID
 and n.NATIONALITY = f.NATIONALITY
 and n.NATIONALITY IN
('Angolan','Australian','British','Bangladeshi','Chinese','Batswana')
 GROUP BY n.NATIONALITY, tr.classtypedesc
 ) where "Percent Rank by nationality" >= 0.9;
```

d) Screenshot of query result



	NATIONALITY	CLASSTYPEDESC	Total Revenue	Percent Rank by nationality	Percent Rank by travel class
1	Angolan	Business Class	11505.23	1	0.2
2	Australian	Business Class	4118629.15	1	1
3	Bangladeshi	Business Class	6405.68	1	0
4	Batswana	First Class	14653.89	1	0.8
5	British	Business Class	18336.08	1	0.8
6	Chinese	Business Class	16249.89	1	0.6

e) Comparison of rank and percent rank

- Rank: It computes the rank of a record compared to other records in the table based on the values of a set of measures. For example, in report 13, the cities in each country are ranked based on service cost.
- Percent rank: Percent rank also computes the ranking of a record but in a percentage form unlike the number form through rank. For example, in report 14, based on total revenue, we have determined the top 10% of the total revenue generated by a particular nationality of passengers, travelling a particular class.

3. Task C.3

3.1.Report 1:

a) SQL Query

```
Select t.YEARNO as Year, s.city as Source_City,
sum(f.TOTALNUMBEROFPASSENGERS) as Total_Number_Of_Passengers
from source_dim s, passtransfact f, time_dim t
where s.SOURCE_ID = f.SOURCEAIRPORTID and
t.TIMEID = f.TIMEID and
s.COUNTRY = 'Australia'
group by t.YEARNO, s.city
order by t.YEARNO, s.city;
```

b) Screenshot of the query result

```
627  --Query 1 Show All
628  --What is the total number of passengers who departed from cities of Australia each year?
629
630  Select t.YEARNO as Year, s.city as Source_City, sum(f.TOTALNUMBEROFPASSENGERS) as Total_Number_Of_Passengers
631  from source_dim s, passtransfact f, time_dim t
632  where s.SOURCE_ID = f.SOURCEAIRPORTID and
633  t.TIMEID = f.TIMEID and
634  s.COUNTRY = 'Australia'
635  group by t.YEARNO, s.city
636  order by t.YEARNO, s.city;
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.241 seconds

	YEAR	SOURCE_CITY	TOTAL_NUMBER_OF_PASSENGERS
1	2007	Adelaide	124
2	2007	Albury	18
3	2007	Alice Springs	55
4	2007	Aurukun	12
5	2007	Blackall	16
6	2007	Brisbane	662
7	2007	Broken Hill	17
8	2007	Broome	30
9	2007	Brusselton	20
10	2007	Burketown	18
11	2007	Burnie	16
12	2007	Cairns	270

c) Execution plan of original query

```
Explain Plan For
Select t.YEARNO as Year, s.city as Source_City,
sum(f.TOTALNUMBEROFPASSENGERS) as Total_Number_Of_Passengers
from source_dim s, passtransfact f, time_dim t
where s.SOURCE_ID = f.SOURCEAIRPORTID and
t.TIMEID = f.TIMEID and
s.COUNTRY = 'Australia'
group by t.YEARNO, s.city
```

order by t.YEARNO, s.city;

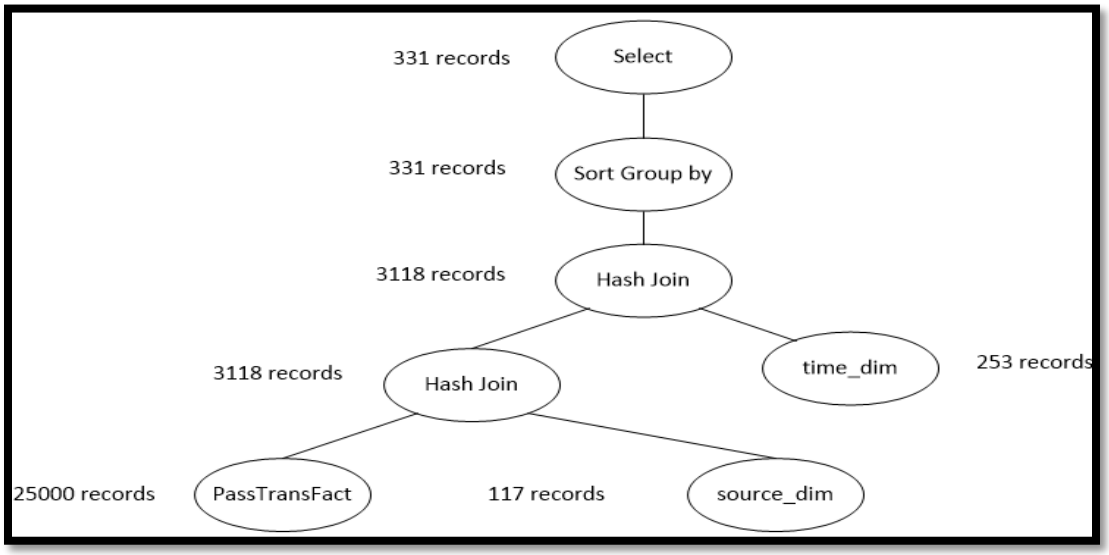
Select * From Table(dbms_xplan.display);

Script Output | Query Result

SQL | All Rows Fetched: 20 in 0.018 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 1512711299

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |               |   331 | 22508 |    97   (4)| 00:00:02 |
|   1 |  SORT GROUP BY       |               |   331 | 22508 |    97   (4)| 00:00:02 |
|*  2 |   HASH JOIN          |               |  3118 |   207K|    96   (3)| 00:00:02 |
|   3 |    TABLE ACCESS FULL | TIME_DIM      |   253 |  5313 |     3   (0)| 00:00:01 |
|*  4 |    HASH JOIN         |               |  3118 |   143K|    92   (2)| 00:00:02 |
|*  5 |     TABLE ACCESS FULL| SOURCE_DIM    |   117 |  2808 |     7   (0)| 00:00:01 |
|   6 |     TABLE ACCESS FULL| PASSTRANSFACT | 25000 |   561K|    84   (0)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

d) Query tree of the original query



e) New SQL

```
Select * from
(Select t.YEARNO as Year, s.city as Source_City,
sum(f.TOTALNUMBEROFPASSENGERS) as Total_Number_Of_Passengers
from source_dim s, passtransfact f, time_dim t
where s.SOURCE_ID = f.SOURCEAIRPORTID and
t.TIMEID = f.TIMEID and
s.COUNTRY = 'Australia'
group by t.YEARNO, s.city )
order by Year, Source_City;
```

f) Screenshot of the new query result

Script Output | Query Result

SQL | Fetched 50 rows in 0.029 seconds

	YEAR	SOURCE_CITY	TOTAL_NUMBER_OF_PASSENGERS
1	2007	Adelaide	124
2	2007	Albury	18
3	2007	Alice Springs	55
4	2007	Aurukun	12
5	2007	Blackall	16
6	2007	Brisbane	662
7	2007	Broken Hill	17
8	2007	Broome	30
9	2007	Brusselton	20
10	2007	Burketown	18
11	2007	Burnie	16
12	2007	Cairns	270

g) Execution plan of new query

```
Explain Plan For
Select * from
    (Select /*+ NO_MERGE */  t.YEARNO as Year, s.city as Source_City,
    sum(f.TOTALNUMBEROFPASSENGERS) as Total_Number_Of_Passengers
    from source_dim s, passtransfact f, time_dim t
    where s.SOURCE_ID = f.SOURCEAIRPORTID and
    t.TIMEID = f.TIMEID and
    s.COUNTRY = 'Australia'
    group by t.YEARNO, s.city )
```

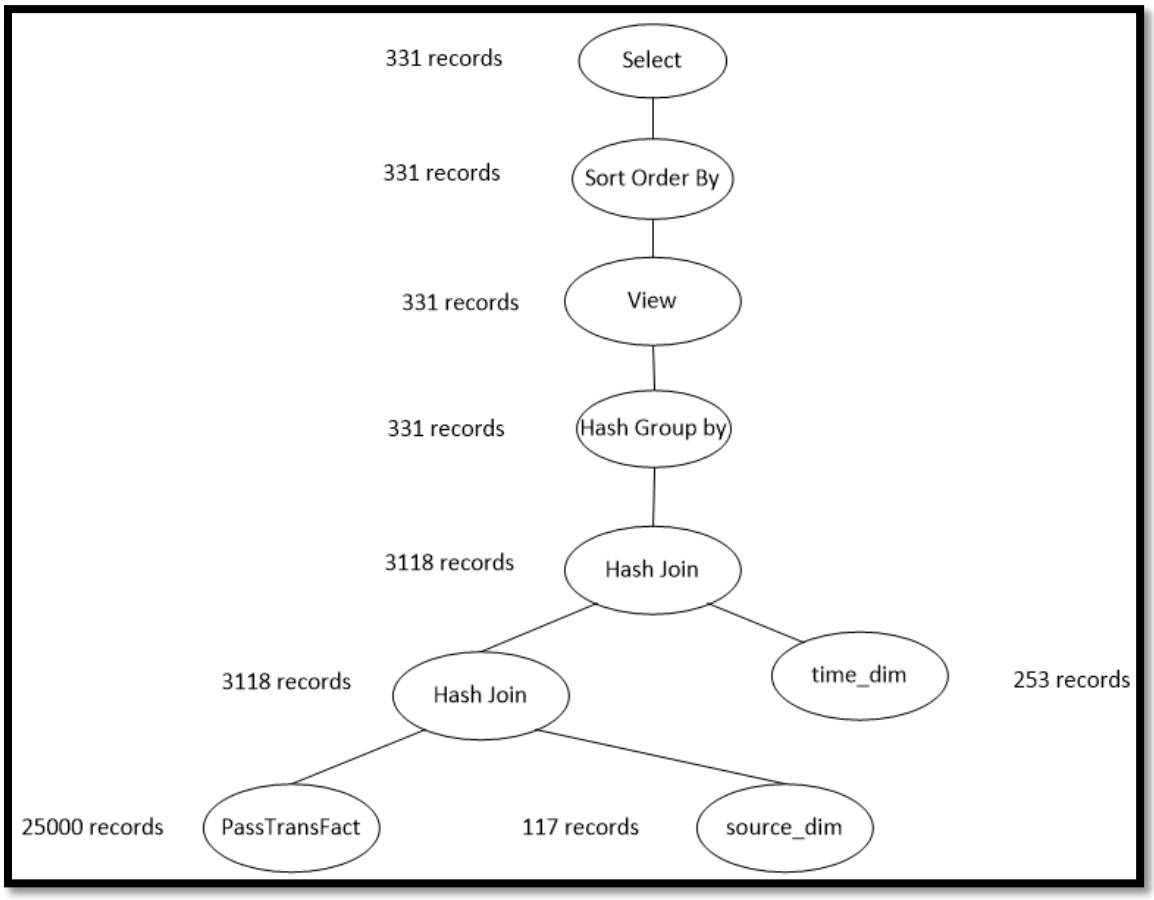
order by Year, Source_City;

Select * From Table(dbms_xplan.display);

Script Output | Query Result | SQL | All Rows Fetched: 22 in 0.022 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 1780134825

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               |   331 | 14564 |    97   (4)| 00:00:02 |
|   1 |  SORT ORDER BY        |               |   331 | 14564 |    97   (4)| 00:00:02 |
|   2 |   VIEW                |               |   331 | 14564 |    97   (4)| 00:00:02 |
|   3 |    HASH GROUP BY      |               |   331 | 22508 |    97   (4)| 00:00:02 |
|*  4 |     HASH JOIN         |               |  3118 |   207K|    96   (3)| 00:00:02 |
|   5 |      TABLE ACCESS FULL| TIME_DIM      |   253 |  5313 |     3   (0)| 00:00:01 |
|*  6 |      HASH JOIN        |               |  3118 |   143K|    92   (2)| 00:00:02 |
|*  7 |       TABLE ACCESS FULL| SOURCE_DIM    |   117 |  2808 |     7   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| PASSTRANSFACT | 25000 |   561K|    84   (0)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

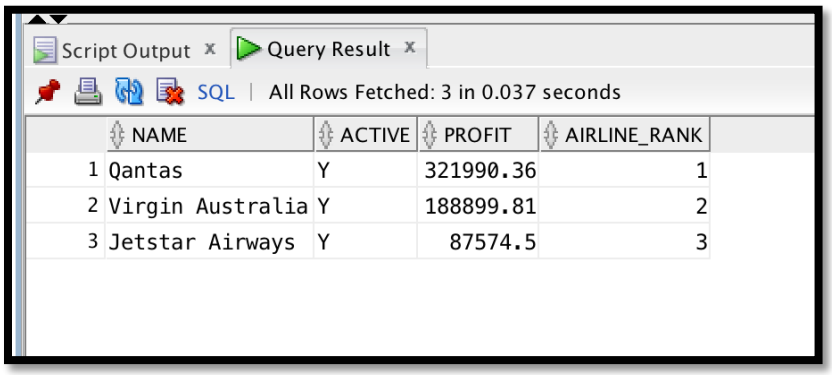
New query is better than old query because it is advisable to use hash group by instead of sort group by when the number of records in a table are in millions. The efficiency of hash group by is more than sort group by as it aggregates large number of rows into aggregates, thus reducing CPU time. We can perform the group by in-line view and perform the order by in the outer query. Use the NO_MERGE hint to prevent the two operations from being combined.

3.2.Report 2:

a) SQL Query

```
SELECT *
FROM
(SELECT a.Name, a.ACTIVE, SUM(f.TOTALPROFIT) AS Profit,
RANK() OVER (ORDER BY SUM(f.TOTALPROFIT) DESC ) AS
Airline_RANK
 FROM passtransfact f, AIRLINE_DIM a, TIME_DIM t
 WHERE f.TIMEID = t.TIMEID
AND f.AIRLINEID = a.AIRLINEID
AND a.Country = 'Australia'
AND t.YEARNO = 2007
GROUP BY a.Name, a.ACTIVE)
WHERE Airline_RANK <= 3;
```

b) Screenshot of the query result



	NAME	ACTIVE	PROFIT	AIRLINE_RANK
1	Qantas	Y	321990.36	1
2	Virgin Australia	Y	188899.81	2
3	Jetstar Airways	Y	87574.5	3

c) Execution plan of original query

```
Explain Plan For
SELECT *
FROM
(SELECT a.Name, a.ACTIVE, SUM(f.TOTALPROFIT) AS Profit,
RANK() OVER (ORDER BY SUM(f.TOTALPROFIT) DESC ) AS
Airline_RANK
FROM passtransfact f, AIRLINE_DIM a, TIME_DIM t
WHERE f.TIMEID = t.TIMEID
AND f.AIRLINEID = a.AIRLINEID
```

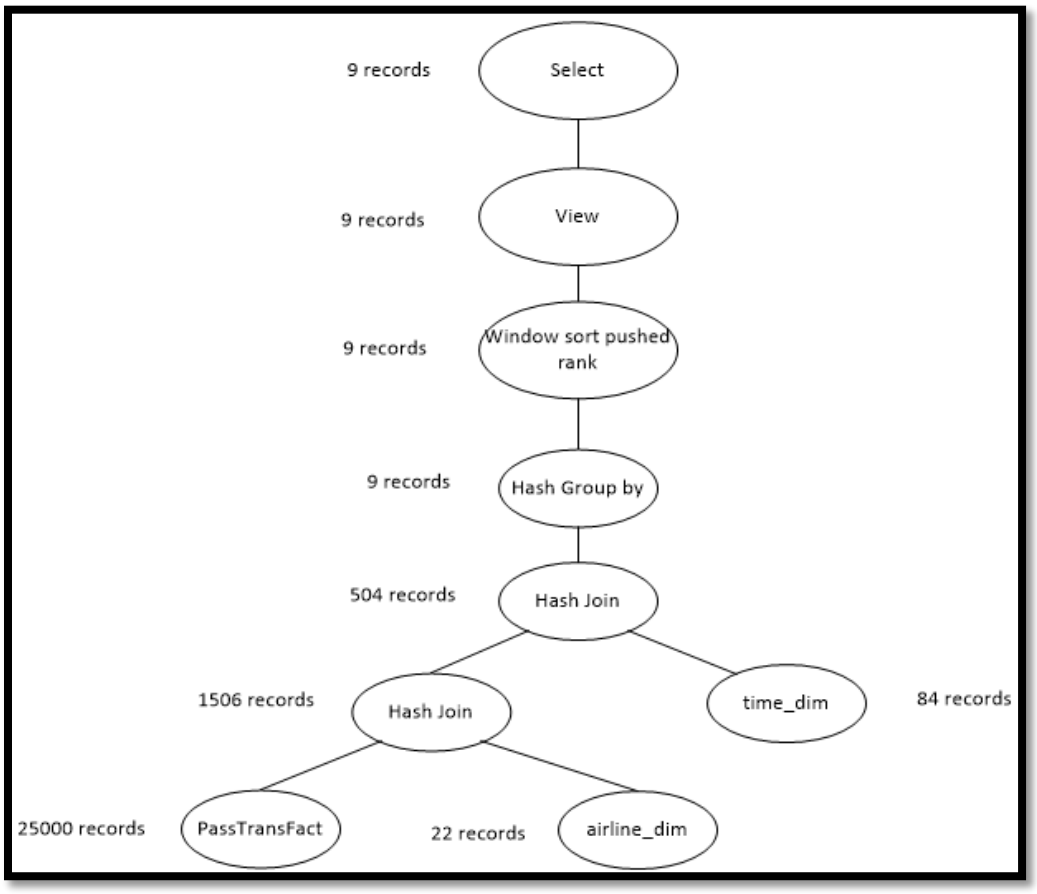
```
AND a.Country = 'Australia'
AND t.YEARNO = 2007
GROUP BY a.Name, a.ACTIVE)
WHERE Airline_RANK <= 3;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 29 in 0.023 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 2202005459

-------------------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |               |     9 |   729 |   105   (4)| 00:00:02 |
|*  1 |  VIEW                    |               |     9 |   729 |   105   (4)| 00:00:02 |
|*  2 |   WINDOW SORT PUSHED RANK|               |     9 |   801 |   105   (4)| 00:00:02 |
|   3 |    HASH GROUP BY         |               |     9 |   801 |   105   (4)| 00:00:02 |
|*  4 |     HASH JOIN            |               |   504 | 44856 |   103   (2)| 00:00:02 |
|*  5 |      TABLE ACCESS FULL   | TIME_DIM      |    84 |  2268 |     3   (0)| 00:00:01 |
|*  6 |      HASH JOIN           |               |  1506 | 93372 |    99   (2)| 00:00:02 |
|*  7 |       TABLE ACCESS FULL  | AIRLINE_DIM   |    22 |   814 |    14   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL  | PASSTRANSFACT | 25000 |   610K|    85   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------------

```

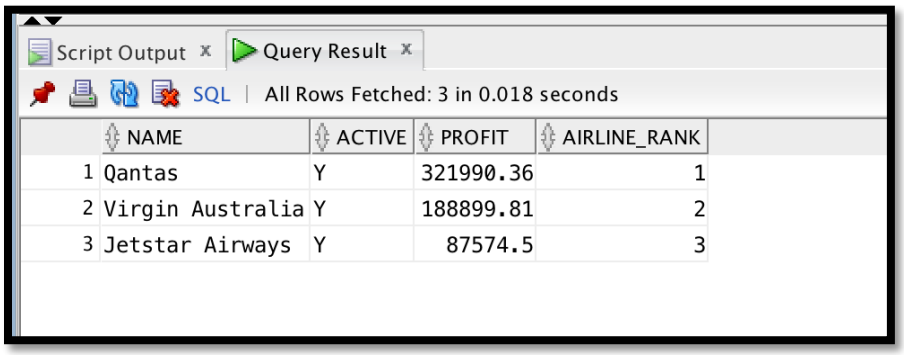
d) Query tree of the original query



e) New SQL

```
SELECT *
FROM
(SELECT /*+ ORDERED */ a.Name, a.ACTIVE, SUM(f.TOTALPROFIT) AS Profit,
RANK() OVER (ORDER BY SUM(f.TOTALPROFIT) DESC ) AS
Airline_RANK
FROM TIME_DIM t, AIRLINE_DIM a, passtransfact f
WHERE f.TIMEID = t.TIMEID
AND f.AIRLINEID = a.AIRLINEID
AND a.Country = 'Australia'
AND t.YEARNO = 2007
GROUP BY a.Name, a.ACTIVE)
WHERE Airline_RANK <= 3;
```

f) Screenshot of the new query result



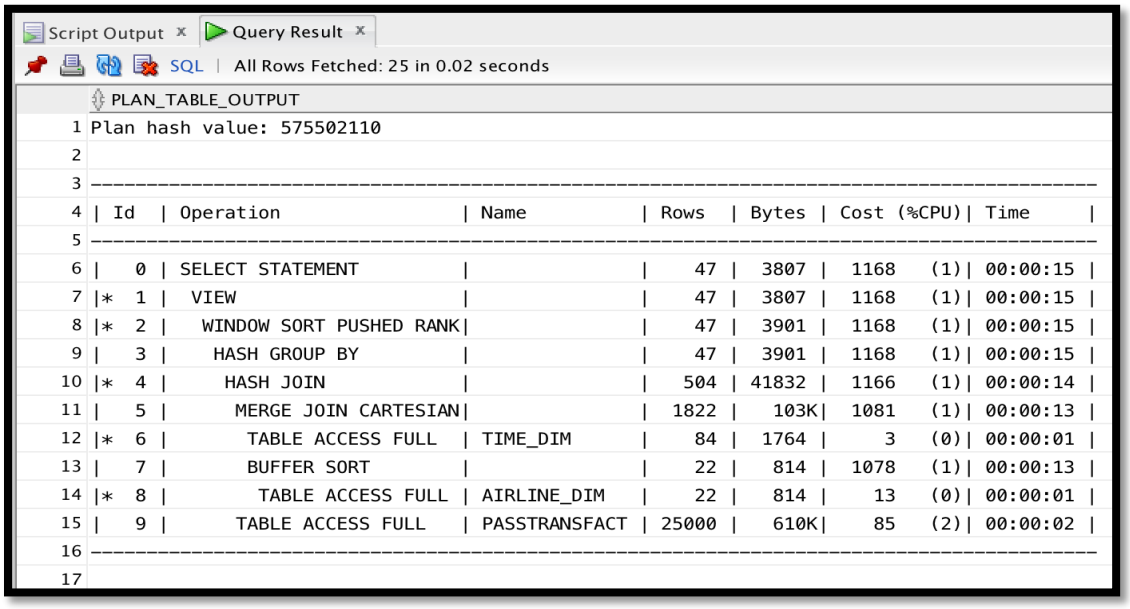
Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.018 seconds

	NAME	ACTIVE	PROFIT	AIRLINE_RANK
1	Qantas	Y	321990.36	1
2	Virgin Australia	Y	188899.81	2
3	Jetstar Airways	Y	87574.5	3

g) Execution plan of new query

```
Explain Plan For
SELECT *
FROM
(SELECT /*+ ORDERED */ a.Name, a.ACTIVE, SUM(f.TOTALPROFIT) AS Profit,
RANK() OVER (ORDER BY SUM(f.TOTALPROFIT) DESC ) AS
Airline_RANK
FROM TIME_DIM t, AIRLINE_DIM a, passtransfact f
WHERE f.TIMEID = t.TIMEID
AND f.AIRLINEID = a.AIRLINEID
AND a.Country = 'Australia'
AND t.YEARNO = 2007
GROUP BY a.Name, a.ACTIVE)
WHERE Airline_RANK <= 3;

Select * From Table(dbms_xplan.display);
```

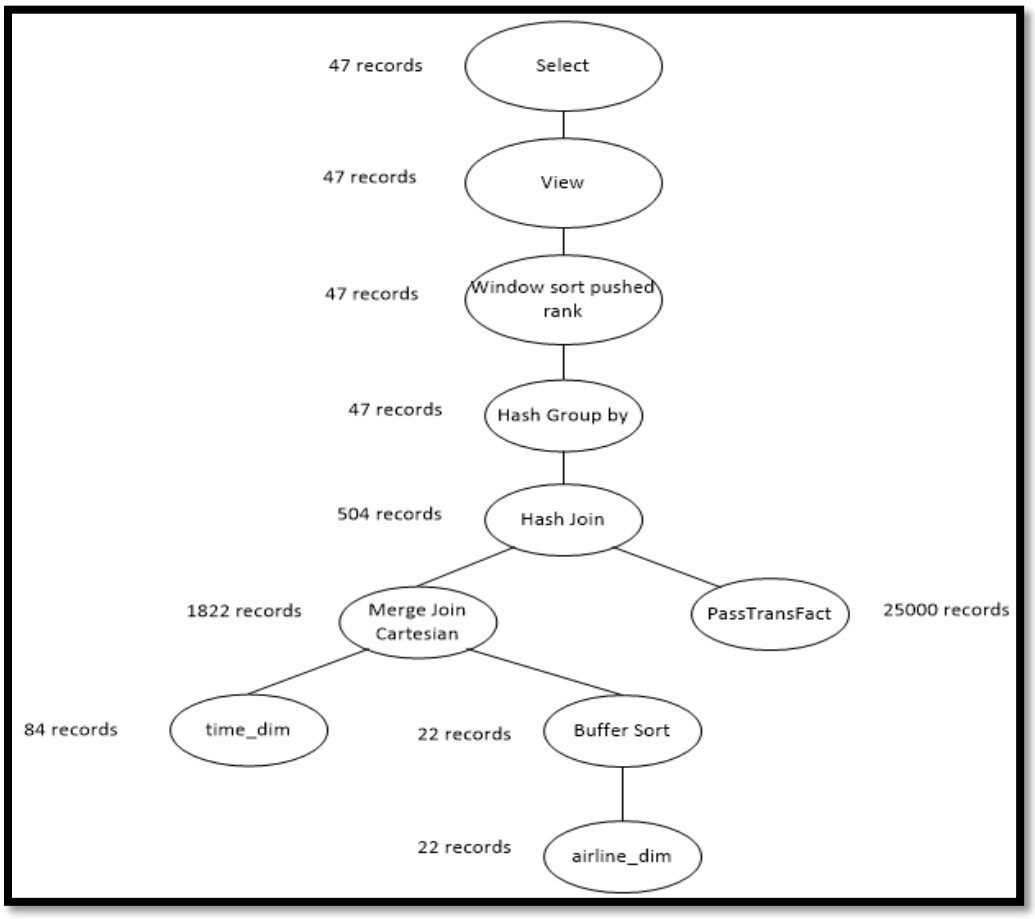


Script Output | Query Result

SQL | All Rows Fetched: 25 in 0.02 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 575502110

-------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |               |    47 |  3807 |  1168   (1)| 00:00:15 |
|*  1 |  VIEW                      |               |    47 |  3807 |  1168   (1)| 00:00:15 |
|*  2 |   WINDOW SORT PUSHED RANK  |               |    47 |  3901 |  1168   (1)| 00:00:15 |
|   3 |    HASH GROUP BY           |               |    47 |  3901 |  1168   (1)| 00:00:15 |
|*  4 |     HASH JOIN              |               |   504 | 41832 |  1166   (1)| 00:00:14 |
|   5 |      MERGE JOIN CARTESIAN  |               |  1822 |   103K|  1081   (1)| 00:00:13 |
|*  6 |       TABLE ACCESS FULL    | TIME_DIM      |    84 |  1764 |     3   (0)| 00:00:01 |
|   7 |       BUFFER SORT          |               |    22 |   814 |  1078   (1)| 00:00:13 |
|*  8 |        TABLE ACCESS FULL   | AIRLINE_DIM   |    22 |   814 |    13   (0)| 00:00:01 |
|   9 |      TABLE ACCESS FULL     | PASSTRANSFACT | 25000 |   610K|    85   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

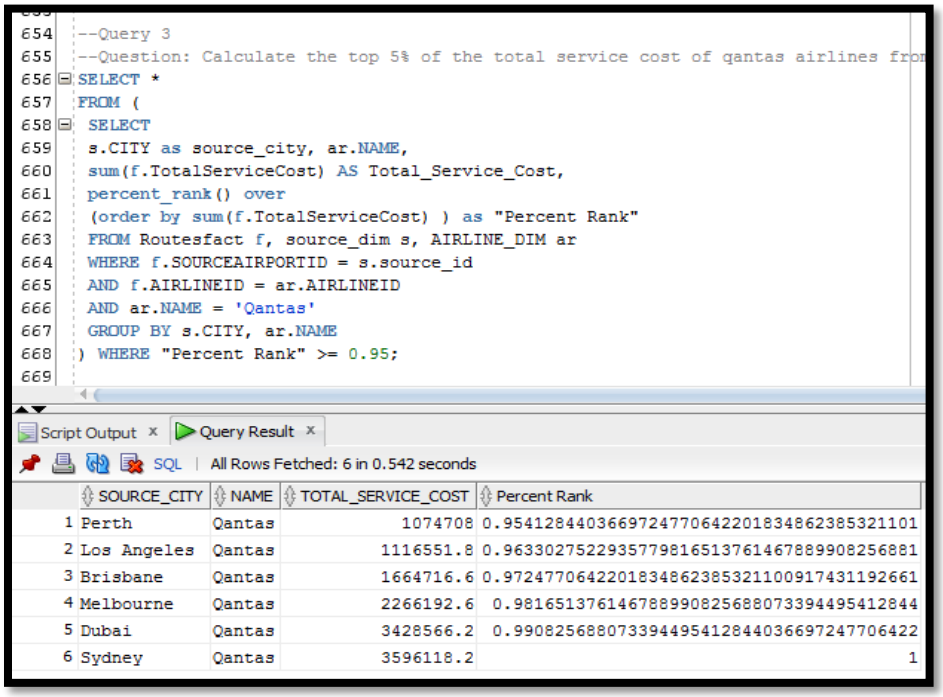
When two dimensions have less number of records, it is advisable to do a Cartesian join between the two so as to prevent the baggage of the fact table records to be carried from the bottom to the top. But in this case, the PassTransFact has maximum records of 25,000 is read at the last stage, so the time of execution and CPU effort is minimized. The original query is more efficient because the hash join of fact table with airline_dim gives 1506 records. So we don't carry the baggage of fact table records till the top.

3.3.Report 3:

a) SQL Query

```
SELECT *
FROM (
SELECT
s.CITY as source_city, ar.NAME,
sum(f.TotalServiceCost) AS Total_Service_Cost,
percent_rank() over
(order by sum(f.TotalServiceCost) ) as "Percent Rank"
FROM Routesfact f, source_dim s, AIRLINE_DIM ar
WHERE f.SOURCEAIRPORTID = s.source_id
AND f.AIRLINEID = ar.AIRLINEID
AND ar.NAME = 'Qantas'
GROUP BY s.CITY, ar.NAME
) WHERE "Percent Rank" >= 0.95;
```

b) Screenshot of the query result



```
654  --Query 3
655  --Question: Calculate the top 5% of the total service cost of qantas airlines fro
656  SELECT *
657  FROM (
658   SELECT
659   s.CITY as source_city, ar.NAME,
660   sum(f.TotalServiceCost) AS Total_Service_Cost,
661   percent_rank() over
662   (order by sum(f.TotalServiceCost) ) as "Percent Rank"
663   FROM Routesfact f, source_dim s, AIRLINE_DIM ar
664   WHERE f.SOURCEAIRPORTID = s.source_id
665   AND f.AIRLINEID = ar.AIRLINEID
666   AND ar.NAME = 'Qantas'
667   GROUP BY s.CITY, ar.NAME
668  ) WHERE "Percent Rank" >= 0.95;
669
```

Script Output | Query Result | SQL | All Rows Fetched: 6 in 0.542 seconds

	SOURCE_CITY	NAME	TOTAL_SERVICE_COST	Percent Rank
1	Perth	Qantas	1074708	0.954128440366972477064220183486238532110 1
2	Los Angeles	Qantas	1116551.8	0.963302752293577981651376146788990825688 1
3	Brisbane	Qantas	1664716.6	0.972477064220183486238532110091743119266 1
4	Melbourne	Qantas	2266192.6	0.981651376146788990825688073394495412844
5	Dubai	Qantas	3428566.2	0.990825688073394495412844036697247706422
6	Sydney	Qantas	3596118.2	1

c) Execution plan of original query

```
Explain Plan For
SELECT *
FROM (
 SELECT
 s.CITY as source_city, ar.NAME,
```

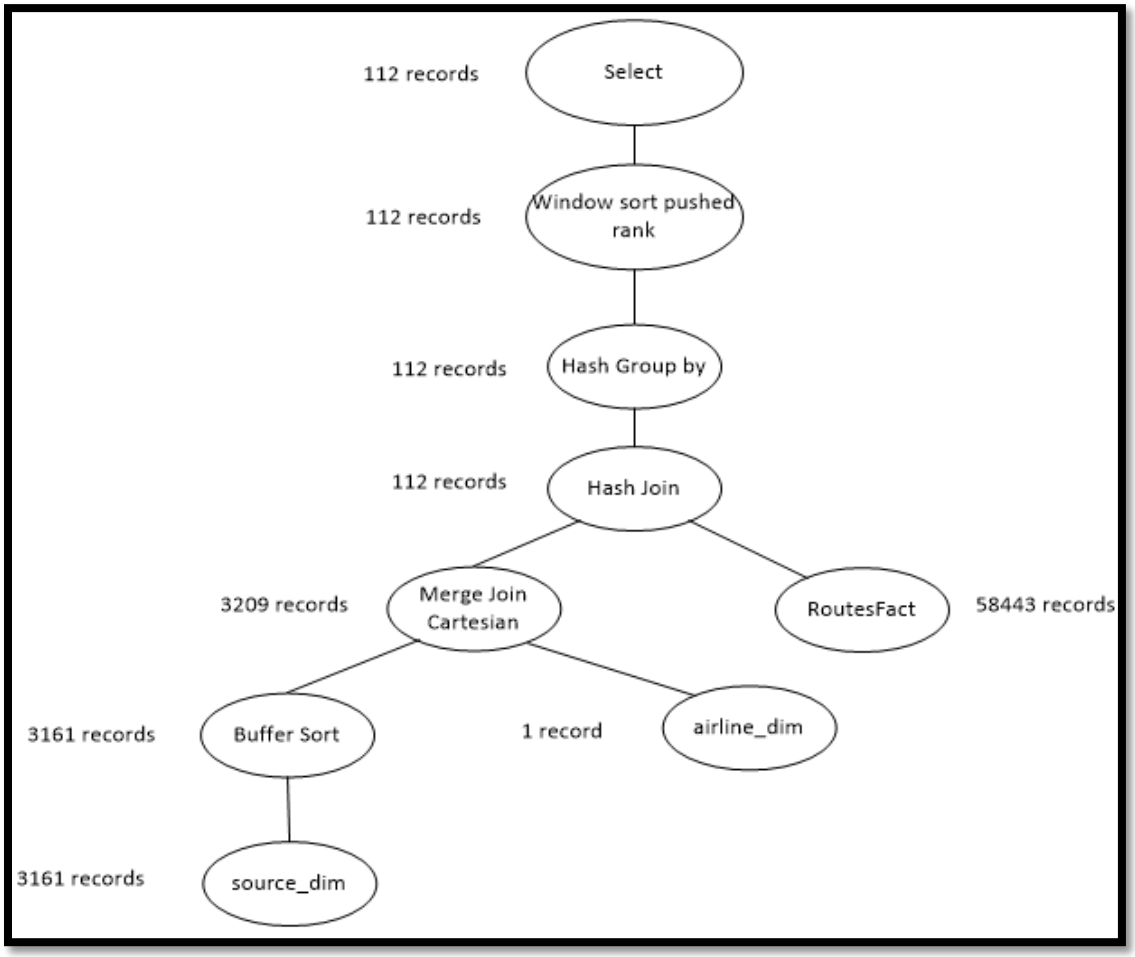
```
 sum(f.TotalServiceCost) AS Total_Service_Cost,
 percent_rank() over
 (order by sum(f.TotalServiceCost) ) as "Percent Rank"
 FROM Routesfact f, source_dim s, AIRLINE_DIM ar
 WHERE f.SOURCEAIRPORTID = s.source_id
 AND f.AIRLINEID = ar.AIRLINEID
 AND ar.NAME = 'Qantas'
 GROUP BY s.CITY, ar.NAME
) WHERE "Percent Rank" >= 0.95;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 24 in 0.021 seconds

	PLAN_TABLE_OUTPUT
1	Plan hash value: 2252191069
2	
3	--
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	--
6	| 0 | SELECT STATEMENT | | 112 | 11760 | 98 (5)| 00:00:02 |
7	|* 1 | VIEW | | 112 | 11760 | 98 (5)| 00:00:02 |
8	| 2 | WINDOW SORT | | 112 | 5824 | 98 (5)| 00:00:02 |
9	| 3 | HASH GROUP BY | | 112 | 5824 | 98 (5)| 00:00:02 |
10	|* 4 | HASH JOIN | | 112 | 5824 | 96 (3)| 00:00:02 |
11	| 5 | MERGE JOIN CARTESIAN| | 3209 | 119K| 21 (0)| 00:00:01 |
12	|* 6 | TABLE ACCESS FULL | AIRLINE_DIM | 1 | 24 | 14 (0)| 00:00:01 |
13	| 7 | BUFFER SORT | | 3161 | 44254 | 7 (0)| 00:00:01 |
14	| 8 | TABLE ACCESS FULL | SOURCE_DIM | 3161 | 44254 | 7 (0)| 00:00:01 |
15	| 9 | TABLE ACCESS FULL | ROUTESFACT | 58443 | 799K| 74 (2)| 00:00:01 |
16	--

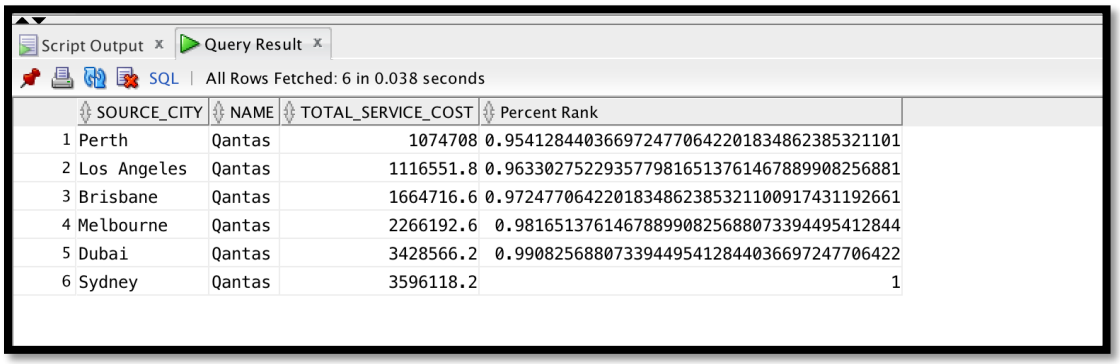
d) Query tree of the original query



e) New SQL

```
SELECT *
FROM (
SELECT /*+ USE_NL (s ar) */
s.CITY as source_city, ar.NAME,
sum(f.TotalServiceCost) AS Total_Service_Cost,
percent_rank() over
(order by sum(f.TotalServiceCost) ) as "Percent Rank"
FROM source_dim s, AIRLINE_DIM ar, Routesfact f
WHERE f.SOURCEAIRPORTID = s.source_id
AND f.AIRLINEID = ar.AIRLINEID
AND ar.NAME = 'Qantas'
GROUP BY s.CITY, ar.NAME
) WHERE "Percent Rank" >= 0.95;
```

f) Query result



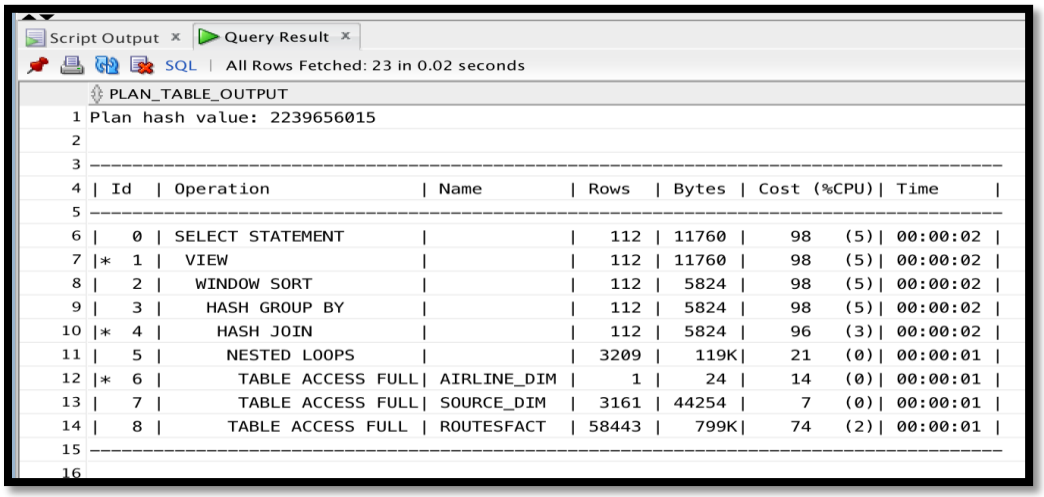
Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.038 seconds

	SOURCE_CITY	NAME	TOTAL_SERVICE_COST	Percent Rank
1	Perth	Qantas	1074708	0.954128440366972477064220183486238532110
2	Los Angeles	Qantas	1116551.8	0.963302752293577981651376146788990825688
3	Brisbane	Qantas	1664716.6	0.972477064220183486238532110091743119266
4	Melbourne	Qantas	2266192.6	0.981651376146788990825688073394495412844
5	Dubai	Qantas	3428566.2	0.990825688073394495412844036697247706422
6	Sydney	Qantas	3596118.2	1

g) Execution plan of new query

```
Explain Plan For
SELECT *
FROM (
 SELECT /*+ USE_NL (s ar) */
 s.CITY as source_city, ar.NAME,
 sum(f.TotalServiceCost) AS Total_Service_Cost,
 percent_rank() over
 (order by sum(f.TotalServiceCost) ) as "Percent Rank"
 FROM source_dim s, AIRLINE_DIM ar, Routesfact f
 WHERE f.SOURCEAIRPORTID = s.source_id
 AND f.AIRLINEID = ar.AIRLINEID
 AND ar.NAME = 'Qantas'
 GROUP BY s.CITY, ar.NAME
) WHERE "Percent Rank" >= 0.95;

Select * From Table(dbms_xplan.display);
```

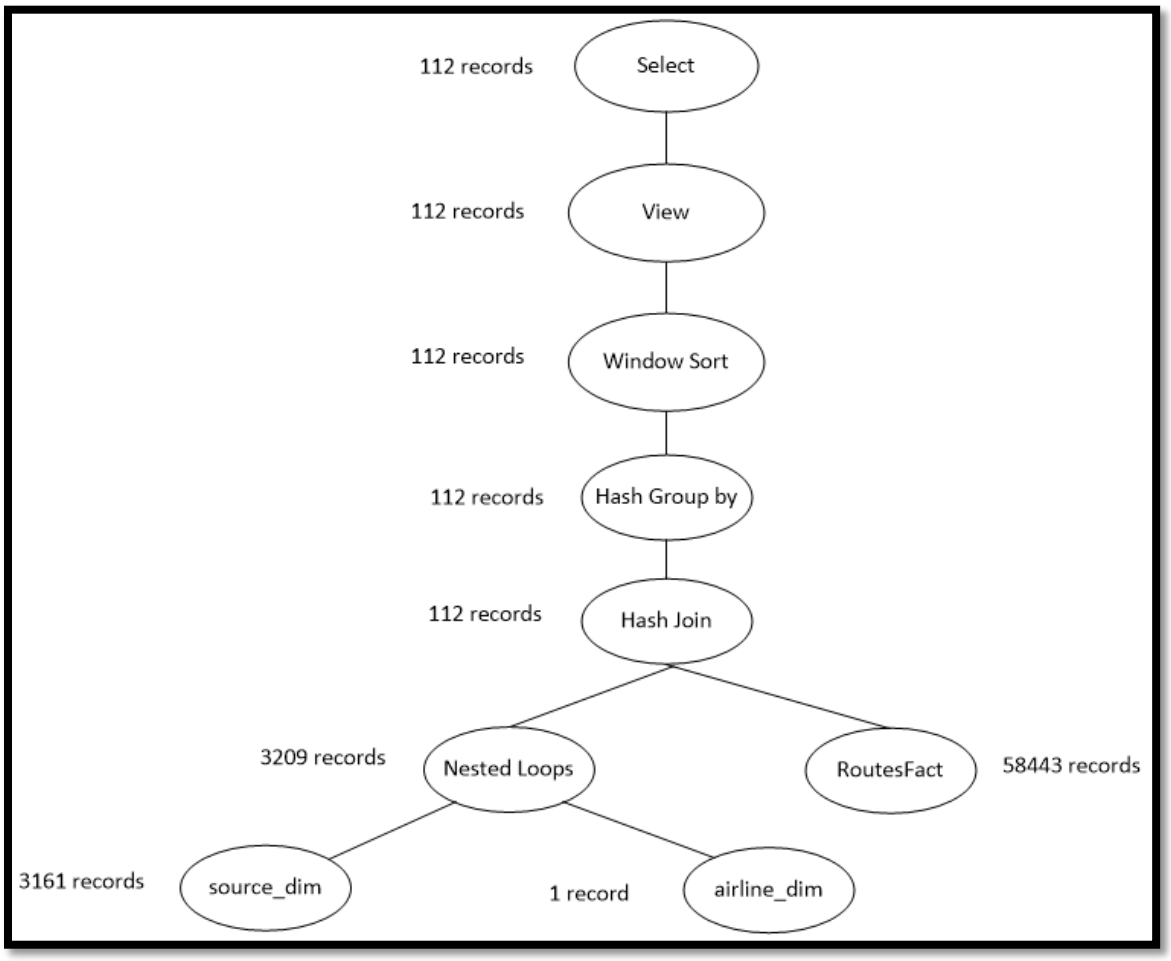


Script Output | Query Result

SQL | All Rows Fetched: 23 in 0.02 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 2239656015

-------------------------------------------------------------------------------------------
| Id  | Operation              | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |              |   112 | 11760 |    98   (5)| 00:00:02 |
|*  1 |  VIEW                  |              |   112 | 11760 |    98   (5)| 00:00:02 |
|   2 |   WINDOW SORT          |              |   112 |  5824 |    98   (5)| 00:00:02 |
|   3 |    HASH GROUP BY       |              |   112 |  5824 |    98   (5)| 00:00:02 |
|*  4 |     HASH JOIN          |              |   112 |  5824 |    96   (3)| 00:00:02 |
|   5 |      NESTED LOOPS      |              |  3209 |  119K|    21   (0)| 00:00:01 |
|*  6 |       TABLE ACCESS FULL| AIRLINE_DIM  |     1 |    24 |    14   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| SOURCE_DIM   |  3161 | 44254 |     7   (0)| 00:00:01 |
|   8 |      TABLE ACCESS FULL | ROUTESFACT   | 58443 |   799K|    74   (2)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

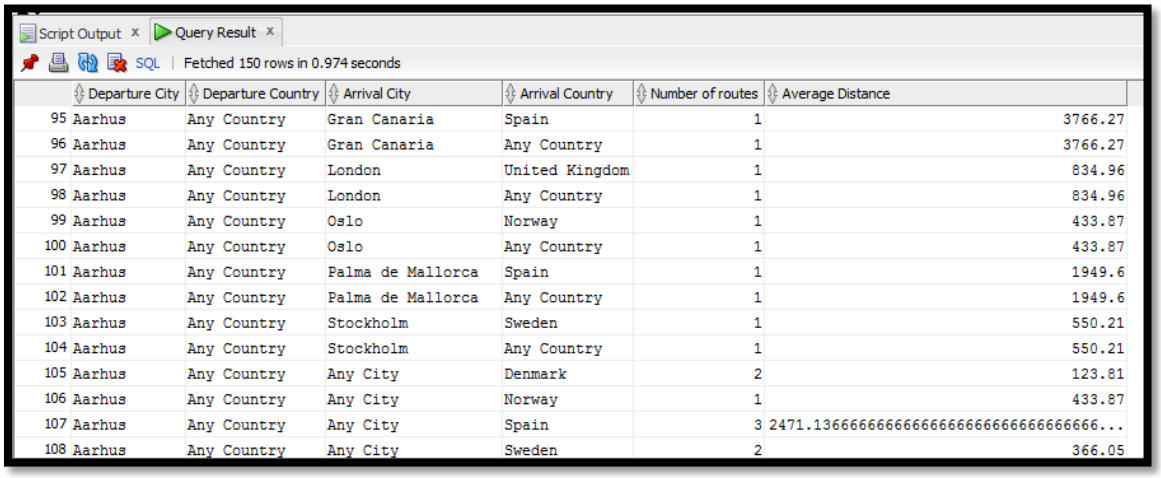
Generally, merge join itself is very fast, but it can be an expensive choice if the number of records in a table are very less. However, if the data volume is large and the desired data can be obtained presorted, merge join is often the fastest available join algorithm. Order by was used while creating source_dim, therefore, we know that the source_dim is already sorted and the number of records are quite large, 3161. Thus, the merge join for airline_dim and sorted source_dim, makes the query more optimized, as it is followed by hash join on the fact table and merge join Cartesian result. On the other side, nested loops is effective if the number of records in a table are less.

3.4.Report 4:

a) SQL Query

```
select decode(grouping(s.CITY),1,'Any City',s.CITY) as "Departure City",
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) as "Departure Country",
decode(grouping(d.CITY),1,'Any City',d.CITY) as "Arrival City",
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) as "Arrival Country",
sum(r.TOTALNUMBEROFROUTES) as "Number of routes",
sum(r.TOTALROUTEDISTANCE)/sum(r.TOTALNUMBEROFROUTES) as "Average
Distance"
from destination_dim d, source_dim s, routesfact r
where d.DESTINATION_ID= r.DESTAIRPORTID
and s.SOURCE_ID = r.SOURCEAIRPORTID
group by cube(s.CITY,s.COUNTRY,d.CITY, d.COUNTRY)
order by s.CITY,s.COUNTRY,d.CITY,d.COUNTRY;
```

b) Screenshot of the query result



Script Output | Query Result | SQL | Fetched 150 rows in 0.974 seconds

	Departure City	Departure Country	Arrival City	Arrival Country	Number of routes	Average Distance
95	Aarhus	Any Country	Gran Canaria	Spain	1	3766.27
96	Aarhus	Any Country	Gran Canaria	Any Country	1	3766.27
97	Aarhus	Any Country	London	United Kingdom	1	834.96
98	Aarhus	Any Country	London	Any Country	1	834.96
99	Aarhus	Any Country	Oslo	Norway	1	433.87
100	Aarhus	Any Country	Oslo	Any Country	1	433.87
101	Aarhus	Any Country	Palma de Mallorca	Spain	1	1949.6
102	Aarhus	Any Country	Palma de Mallorca	Any Country	1	1949.6
103	Aarhus	Any Country	Stockholm	Sweden	1	550.21
104	Aarhus	Any Country	Stockholm	Any Country	1	550.21
105	Aarhus	Any Country	Any City	Denmark	2	123.81
106	Aarhus	Any Country	Any City	Norway	1	433.87
107	Aarhus	Any Country	Any City	Spain	3	2471.13666666666666666666666666666666...
108	Aarhus	Any Country	Any City	Sweden	2	366.05

c) Execution plan of original query

```
Explain Plan For
select decode(grouping(s.CITY),1,'Any City',s.CITY) as "Departure City",
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) as "Departure Country",
decode(grouping(d.CITY),1,'Any City',d.CITY) as "Arrival City",
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) as "Arrival Country",
sum(r.TOTALNUMBEROFROUTES) as "Number of routes",
sum(r.TOTALROUTEDISTANCE)/sum(r.TOTALNUMBEROFROUTES) as "Average
Distance"
from destination_dim d, source_dim s, routesfact r
where d.DESTINATION_ID= r.DESTAIRPORTID
and s.SOURCE_ID = r.SOURCEAIRPORTID
group by cube(s.CITY,s.COUNTRY,d.CITY, d.COUNTRY)
order by s.CITY,s.COUNTRY,d.CITY,d.COUNTRY;
```

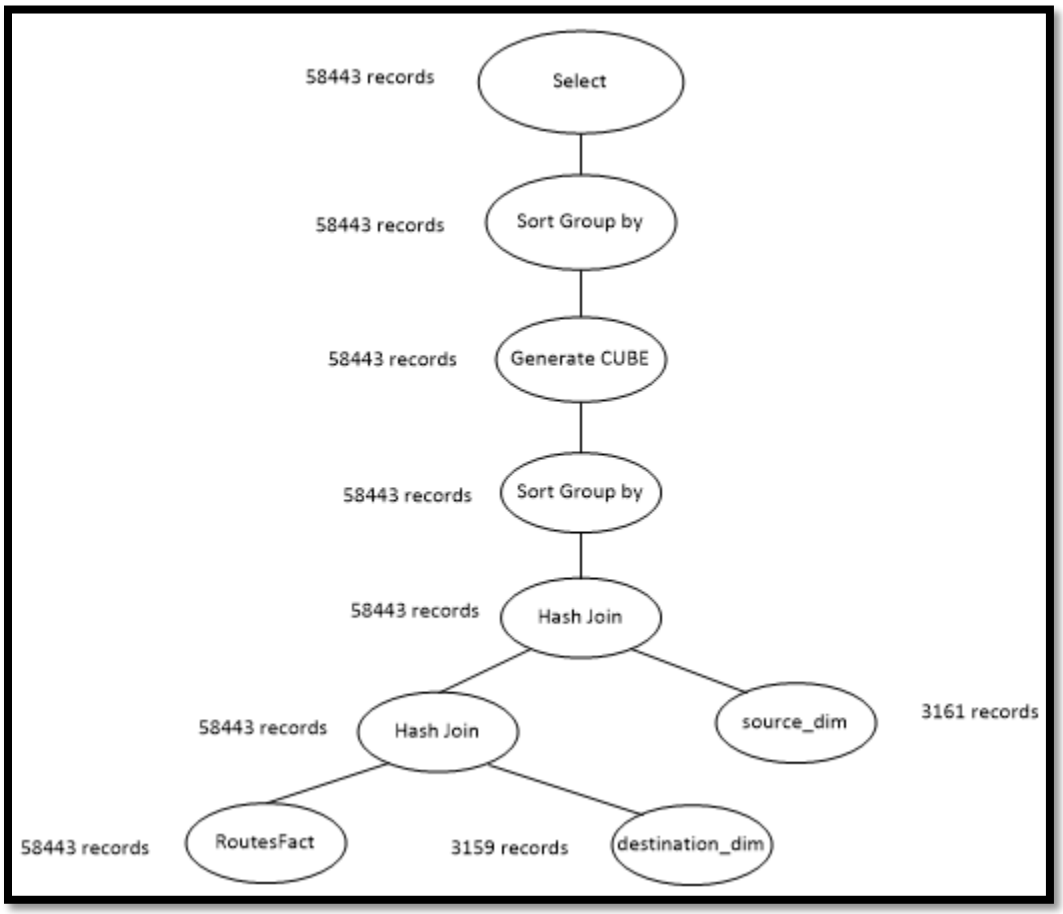
Select * From Table(dbms_xplan.display);

Script Output | Query Result

SQL | All Rows Fetched: 21 in 0.021 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 1188523404

-----------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name            | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                 | 58443 |  3652K|       |   989   (1)| 00:00:12 |
|   1 |  SORT GROUP BY         |                 | 58443 |  3652K|       |   989   (1)| 00:00:12 |
|   2 |   GENERATE CUBE        |                 | 58443 |  3652K|       |   989   (1)| 00:00:12 |
|   3 |    SORT GROUP BY       |                 | 58443 |  3652K|  4384K|   989   (1)| 00:00:12 |
|*  4 |     HASH JOIN          |                 | 58443 |  3652K|       |    89   (3)| 00:00:02 |
|   5 |      TABLE ACCESS FULL | SOURCE_DIM      |  3161 | 75864 |       |     7   (0)| 00:00:01 |
|*  6 |      HASH JOIN         |                 | 58443 |  2282K|       |    82   (3)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| DESTINATION_DIM |  3159 | 75816 |       |     7   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| ROUTESFACT      | 58443 |   913K|       |    74   (2)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------
```

d) Query tree of the original query



e) New SQL

```
SELECT *
FROM (
SELECT /*+ USE_NL (s ar) */
s.CITY as source_city, ar.NAME,
sum(f.TotalServiceCost) AS Total_Service_Cost,
percent_rank() over
(order by sum(f.TotalServiceCost) ) as "Percent Rank"
FROM source_dim s, AIRLINE_DIM ar, Routesfact f
WHERE f.SOURCEAIRPORTID = s.source_id
AND f.AIRLINEID = ar.AIRLINEID
AND ar.NAME = 'Qantas'
GROUP BY s.CITY, ar.NAME
) WHERE "Percent Rank" >= 0.95;
```

f) Screenshot of the new query result

Script Output | Query Result

SQL | Fetched 150 rows in 1.008 seconds

	Departure City	Departure Country	Arrival City	Arrival Country	Number of routes	Average Distance
77	Aarhus	Any Country	Gran Canaria	Any Country	1	3766.27
78	Aarhus	Any Country	Gran Canaria	Spain	1	3766.27
79	Aarhus	Any Country	London	Any Country	1	834.96
80	Aarhus	Any Country	London	United Kingdom	1	834.96
81	Aarhus	Any Country	Oslo	Any Country	1	433.87
82	Aarhus	Any Country	Oslo	Norway	1	433.87
83	Aarhus	Any Country	Palma de Mallorca	Any Country	1	1949.6
84	Aarhus	Any Country	Palma de Mallorca	Spain	1	1949.6
85	Aarhus	Any Country	Stockholm	Any Country	1	550.21
86	Aarhus	Any Country	Stockholm	Sweden	1	550.21

g) Execution plan of new query

```
Alter table source_dim
ADD constraint S_PK PRIMARY KEY(source_id);

Explain Plan For
select /*+ INDEX (s S_PK) */decode(grouping(s.CITY),1,'Any City',s.CITY) as "Departure
City",
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) as "Departure Country",
decode(grouping(d.CITY),1,'Any City',d.CITY) as "Arrival City",
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) as "Arrival Country",
sum(r.TOTALNUMBEROFROUTES) as "Number of routes",
sum(r.TOTALROUTEDISTANCE)/sum(r.TOTALNUMBEROFROUTES) as "Average
Distance"
from destination_dim d, source_dim s, routesfact r
```

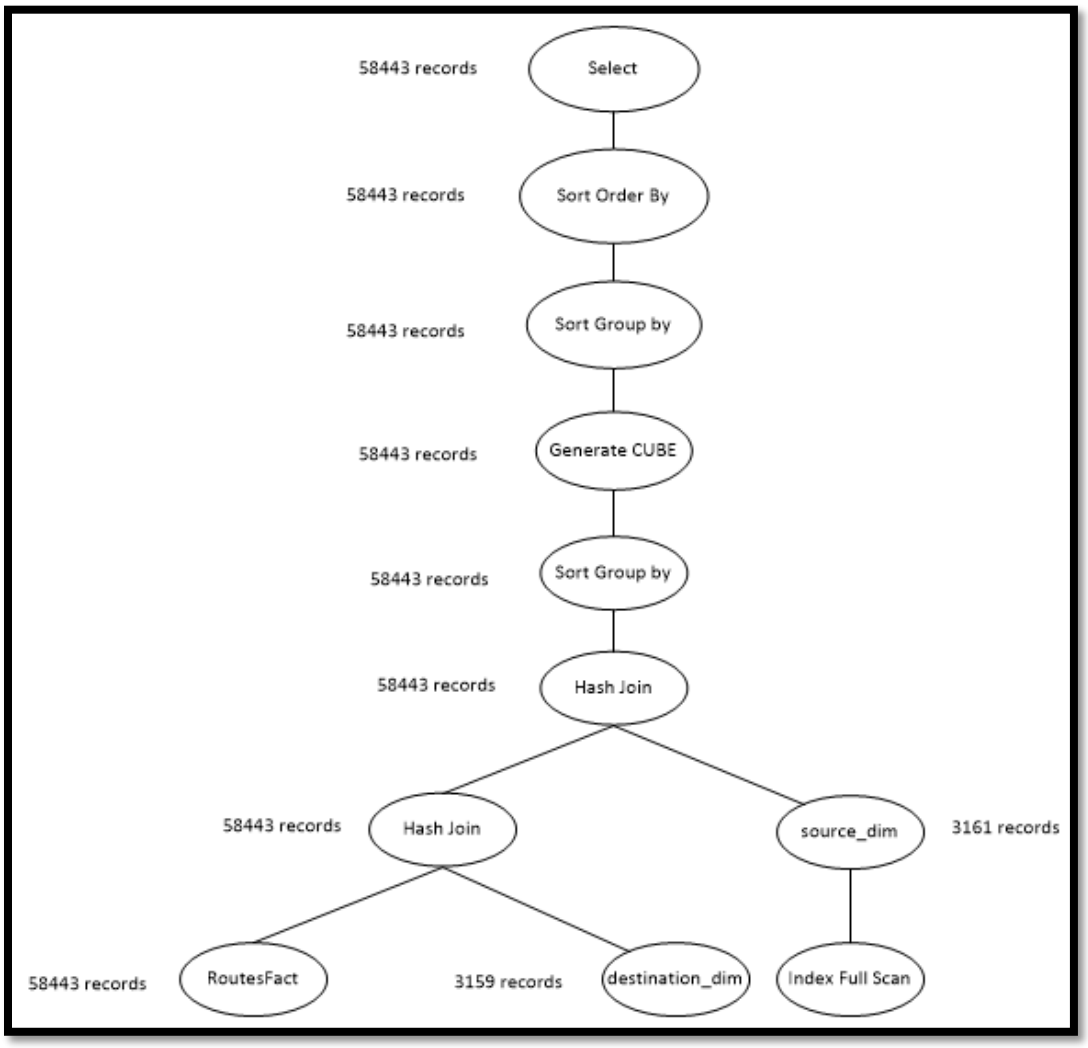
```
where d.DESTINATION_ID= r.DESTAIRPORTID
and s.SOURCE_ID = r.SOURCEAIRPORTID
group by cube(s.CITY,s.COUNTRY,d.CITY, d.COUNTRY)
order by s.CITY,s.COUNTRY,d.CITY,d.COUNTRY;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 23 in 0.018 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 1160347294

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name            | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                 | 58443 |  3652K|       |  1904   (1)| 00:00:23 |
|   1 |  SORT ORDER BY                 |                 | 58443 |  3652K|  4384K|  1904   (1)| 00:00:23 |
|   2 |   SORT GROUP BY                |                 | 58443 |  3652K|       |  1904   (1)| 00:00:23 |
|   3 |    GENERATE CUBE               |                 | 58443 |  3652K|       |  1904   (1)| 00:00:23 |
|   4 |     SORT GROUP BY              |                 | 58443 |  3652K|  4384K|  1904   (1)| 00:00:23 |
|*  5 |      HASH JOIN                 |                 | 58443 |  3652K|       |   104   (2)| 00:00:02 |
|   6 |       TABLE ACCESS BY INDEX ROWID| SOURCE_DIM    |  3161 | 75864 |       |    22   (0)| 00:00:01 |
|   7 |        INDEX FULL SCAN         | S_PK            |  3161 |       |       |     8   (0)| 00:00:01 |
|*  8 |       HASH JOIN                |                 | 58443 |  2282K|       |    82   (3)| 00:00:01 |
|   9 |        TABLE ACCESS FULL       | DESTINATION_DIM |  3159 | 75816 |       |     7   (0)| 00:00:01 |
|  10 |        TABLE ACCESS FULL       | ROUTESFACT      | 58443 |   913K|       |    74   (2)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

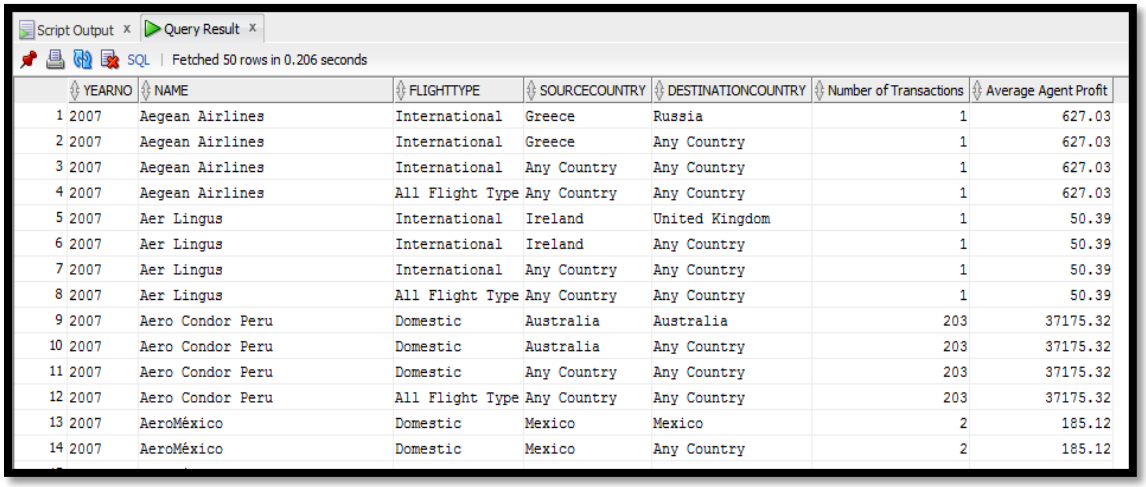
The source_dim table is altered to add source_id as a primary key. The hint of using the primary key as Index was given to the system to alter the execution of the query. However, there is no need to sort the source_dim table, as it is already sorted. Therefore index full scan is an extra step in the execution because of the introduction of the primary key. The original query is optimized and more efficient than the new query.

3.5.Report 5:

a) SQL Query

```
SELECT t.YEARNO, ar.NAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
sum(f.TOTALPROFIT) AS "Average Agent Profit"
FROM passtransfact f, time_dim t, airline_dim ar, Flight_Type_dim ft, source_dim s,
destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.AIRLINEID = ar.AIRLINEID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
GROUP by  t.YEARNO, ar.NAME, rollup(ft.FLIGHTTYPEDESC, s.COUNTRY,
d.COUNTRY)
order by t.YEARNO, ar.NAME, ft.FLIGHTTYPEDESC, s.COUNTRY, d.COUNTRY;
```

b) Screenshot of the query result



Script Output | Query Result | SQL | Fetched 50 rows in 0.206 seconds

	YEARNO	NAME	FLIGHTTYPE	SOURCECOUNTRY	DESTINATIONCOUNTRY	Number of Transactions	Average Agent Profit
1	2007	Aegean Airlines	International	Greece	Russia	1	627.03
2	2007	Aegean Airlines	International	Greece	Any Country	1	627.03
3	2007	Aegean Airlines	International	Any Country	Any Country	1	627.03
4	2007	Aegean Airlines	All Flight Type	Any Country	Any Country	1	627.03
5	2007	Aer Lingus	International	Ireland	United Kingdom	1	50.39
6	2007	Aer Lingus	International	Ireland	Any Country	1	50.39
7	2007	Aer Lingus	International	Any Country	Any Country	1	50.39
8	2007	Aer Lingus	All Flight Type	Any Country	Any Country	1	50.39
9	2007	Aero Condor Peru	Domestic	Australia	Australia	203	37175.32
10	2007	Aero Condor Peru	Domestic	Australia	Any Country	203	37175.32
11	2007	Aero Condor Peru	Domestic	Any Country	Any Country	203	37175.32
12	2007	Aero Condor Peru	All Flight Type	Any Country	Any Country	203	37175.32
13	2007	AeroMéxico	Domestic	Mexico	Mexico	2	185.12
14	2007	AeroMéxico	Domestic	Mexico	Any Country	2	185.12

c) Execution plan of original query

```
Explain Plan For
SELECT t.YEARNO, ar.NAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
```

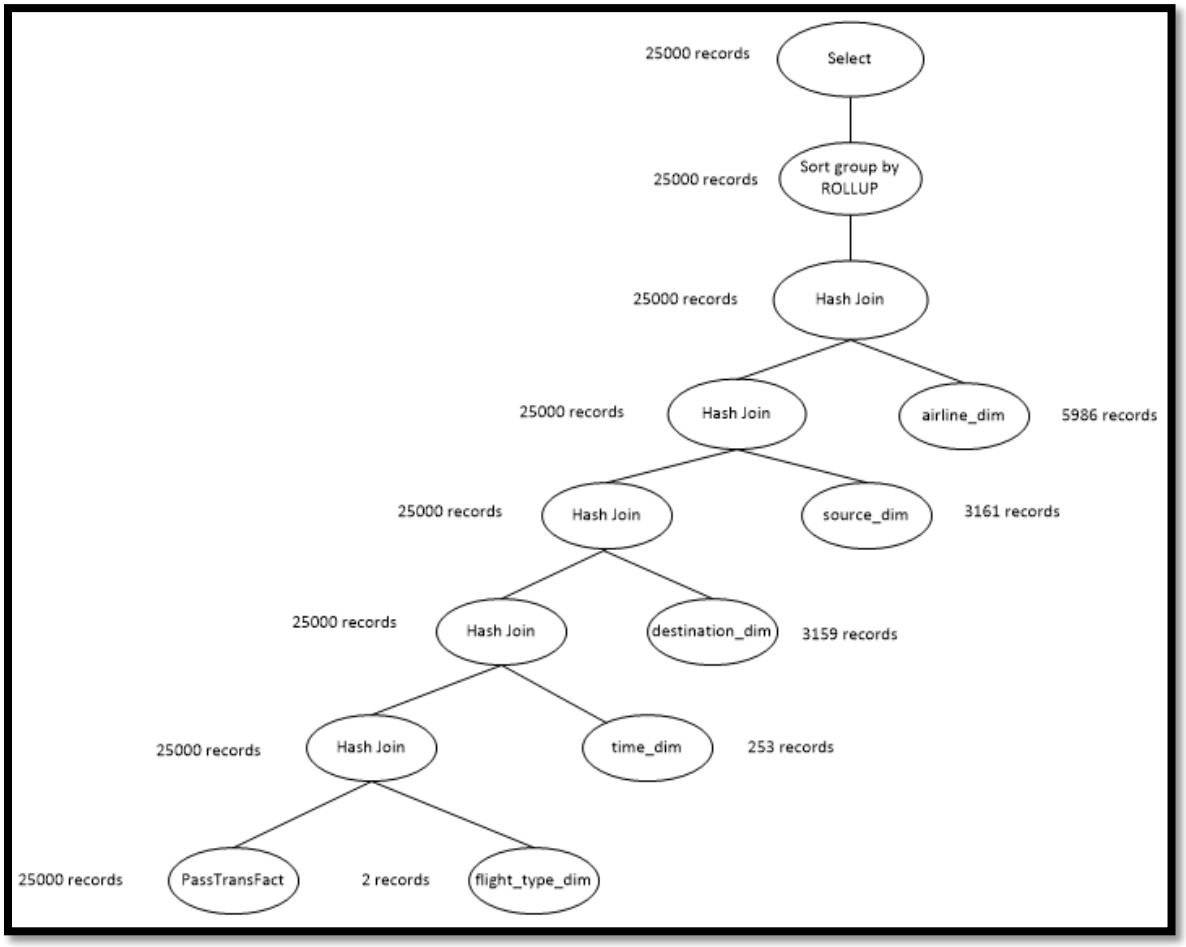
```
sum(f.TOTALPROFIT) AS "Average Agent Profit"
FROM passtransfact f, time_dim t, airline_dim ar, Flight_Type_dim ft, source_dim s,
destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.AIRLINEID = ar.AIRLINEID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
GROUP by  t.YEARNO, ar.NAME, rollup(ft.FLIGHTTYPEDESC, s.COUNTRY,
d.COUNTRY)
order by t.YEARNO, ar.NAME, ft.FLIGHTTYPEDESC, s.COUNTRY, d.COUNTRY;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 32 in 0.032 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 2573695774

-------------------------------------------------------------------------------------------------------
| Id  | Operation               | Name            | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                 | 25000 |  3247K|       |   867   (1)| 00:00:11 |
|   1 |  SORT GROUP BY ROLLUP   |                 | 25000 |  3247K|  3584K|   867   (1)| 00:00:11 |
|*  2 |   HASH JOIN             |                 | 25000 |  3247K|       |   122   (4)| 00:00:02 |
|   3 |    TABLE ACCESS FULL    | AIRLINE_DIM     |  5986 |   140K|       |    14   (0)| 00:00:01 |
|*  4 |    HASH JOIN            |                 | 25000 |  2661K|       |   107   (3)| 00:00:02 |
|   5 |     TABLE ACCESS FULL   | SOURCE_DIM      |  3161 | 44254 |       |     7   (0)| 00:00:01 |
|*  6 |     HASH JOIN           |                 | 25000 |  2319K|       |    99   (3)| 00:00:02 |
|   7 |      TABLE ACCESS FULL  | DESTINATION_DIM |  3159 | 44226 |       |     7   (0)| 00:00:01 |
|*  8 |      HASH JOIN          |                 | 25000 |  1977K|       |    92   (3)| 00:00:02 |
|   9 |       TABLE ACCESS FULL | TIME_DIM        |   253 |  6831 |       |     3   (0)| 00:00:01 |
|* 10 |       HASH JOIN         |                 | 25000 |  1318K|       |    88   (2)| 00:00:02 |
|  11 |        TABLE ACCESS FULL| FLIGHT_TYPE_DIM |     2 |    30 |       |     3   (0)| 00:00:01 |
|  12 |        TABLE ACCESS FULL| PASSTRANSFACT   | 25000 |   952K|       |    85   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------------------
```

d) Query tree of the original query

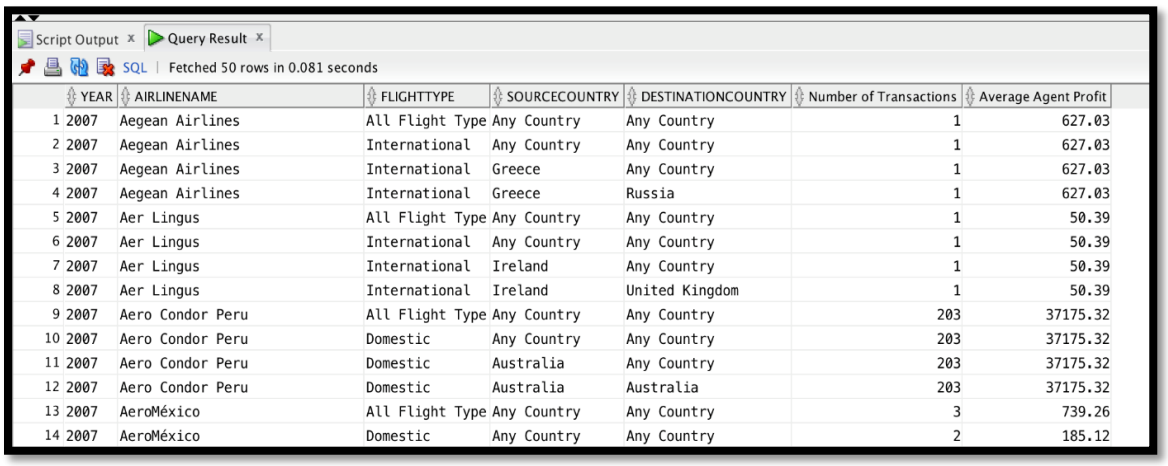


e) New SQL

```
SELECT /*+ USE_HASH_AGGREGATION */t.YEARNO AS Year, ar.NAME AS
AirlineName,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
sum(f.TOTALPROFIT) AS "Average Agent Profit"
FROM passtransfact f, time_dim t, airline_dim ar, Flight_Type_dim ft, source_dim s,
destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.AIRLINEID = ar.AIRLINEID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
```

```
GROUP by  t.YEARNO, ar.NAME, rollup(ft.FLIGHTTYPEDESC, s.COUNTRY,
d.COUNTRY)
order by Year, AirlineName, FlightType, SourceCountry, DestinationCountry;
```

f) Screenshot of the new query result

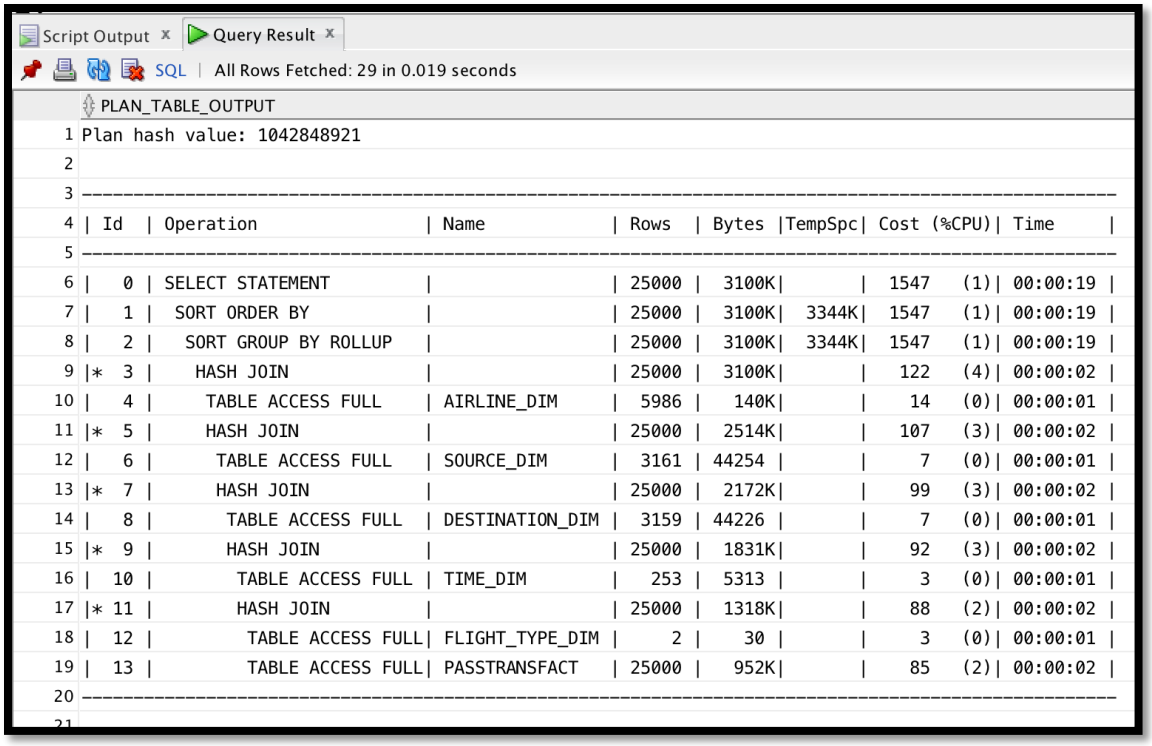


Script Output | Query Result | SQL | Fetched 50 rows in 0.081 seconds

	YEAR	AIRLINENAME	FLIGHTTYPE	SOURCECOUNTRY	DESTINATIONCOUNTRY	Number of Transactions	Average Agent Profit
1	2007	Aegean Airlines	All Flight Type	Any Country	Any Country	1	627.03
2	2007	Aegean Airlines	International	Any Country	Any Country	1	627.03
3	2007	Aegean Airlines	International	Greece	Any Country	1	627.03
4	2007	Aegean Airlines	International	Greece	Russia	1	627.03
5	2007	Aer Lingus	All Flight Type	Any Country	Any Country	1	50.39
6	2007	Aer Lingus	International	Any Country	Any Country	1	50.39
7	2007	Aer Lingus	International	Ireland	Any Country	1	50.39
8	2007	Aer Lingus	International	Ireland	United Kingdom	1	50.39
9	2007	Aero Condor Peru	All Flight Type	Any Country	Any Country	203	37175.32
10	2007	Aero Condor Peru	Domestic	Any Country	Any Country	203	37175.32
11	2007	Aero Condor Peru	Domestic	Australia	Any Country	203	37175.32
12	2007	Aero Condor Peru	Domestic	Australia	Australia	203	37175.32
13	2007	AeroMéxico	All Flight Type	Any Country	Any Country	3	739.26
14	2007	AeroMéxico	Domestic	Any Country	Any Country	2	185.12

g) Execution plan of new query

```
alter table source_dim drop constraint S_PK;

Explain Plan For
SELECT /*+ USE_HASH_AGGREGATION */t.YEARNO AS Year, ar.NAME AS
AirlineName,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
sum(f.TOTALPROFIT) AS "Average Agent Profit"
FROM passtransfact f, time_dim t, airline_dim ar, Flight_Type_dim ft, source_dim s,
destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.AIRLINEID = ar.AIRLINEID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
GROUP by  t.YEARNO, ar.NAME, rollup(ft.FLIGHTTYPEDESC, s.COUNTRY,
d.COUNTRY)
order by Year, AirlineName, FlightType, SourceCountry, DestinationCountry;
```

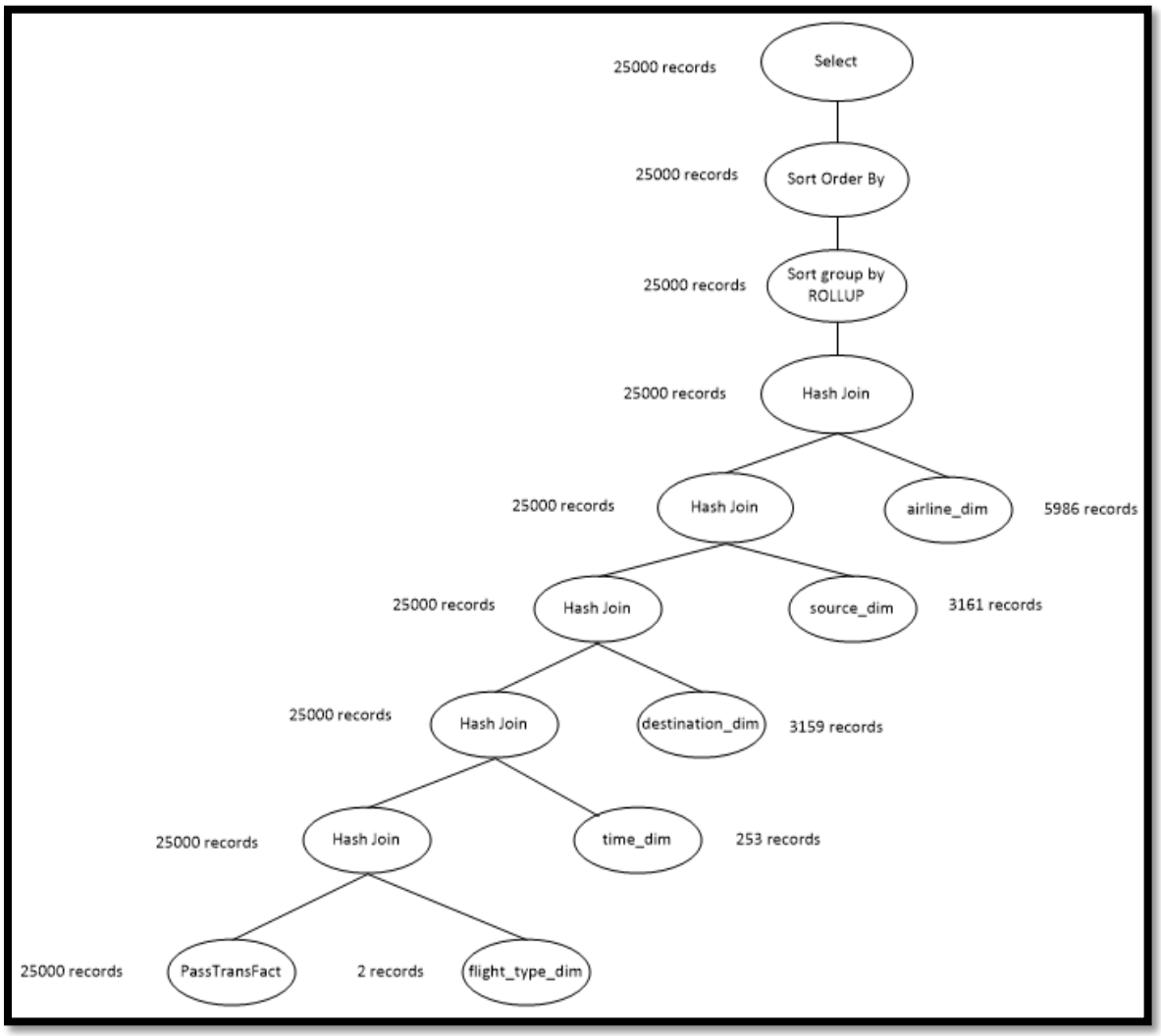


Script Output | Query Result

SQL | All Rows Fetched: 29 in 0.019 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 1042848921

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name             | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                  | 25000 |  3100K|       |  1547   (1)| 00:00:19 |
|   1 |  SORT ORDER BY           |                  | 25000 |  3100K|  3344K|  1547   (1)| 00:00:19 |
|   2 |   SORT GROUP BY ROLLUP   |                  | 25000 |  3100K|  3344K|  1547   (1)| 00:00:19 |
|*  3 |    HASH JOIN             |                  | 25000 |  3100K|       |   122   (4)| 00:00:02 |
|   4 |     TABLE ACCESS FULL    | AIRLINE_DIM      |  5986 |   140K|       |    14   (0)| 00:00:01 |
|*  5 |     HASH JOIN            |                  | 25000 |  2514K|       |   107   (3)| 00:00:02 |
|   6 |      TABLE ACCESS FULL   | SOURCE_DIM       |  3161 | 44254 |       |     7   (0)| 00:00:01 |
|*  7 |      HASH JOIN           |                  | 25000 |  2172K|       |    99   (3)| 00:00:02 |
|   8 |       TABLE ACCESS FULL  | DESTINATION_DIM  |  3159 | 44226 |       |     7   (0)| 00:00:01 |
|*  9 |       HASH JOIN          |                  | 25000 |  1831K|       |    92   (3)| 00:00:02 |
|  10 |        TABLE ACCESS FULL | TIME_DIM         |   253 |  5313 |       |     3   (0)| 00:00:01 |
|* 11 |        HASH JOIN         |                  | 25000 |  1318K|       |    88   (2)| 00:00:02 |
|  12 |         TABLE ACCESS FULL| FLIGHT_TYPE_DIM  |     2 |    30 |       |     3   (0)| 00:00:01 |
|  13 |         TABLE ACCESS FULL| PASSTRANSFACT    | 25000 |  952K|       |    85   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

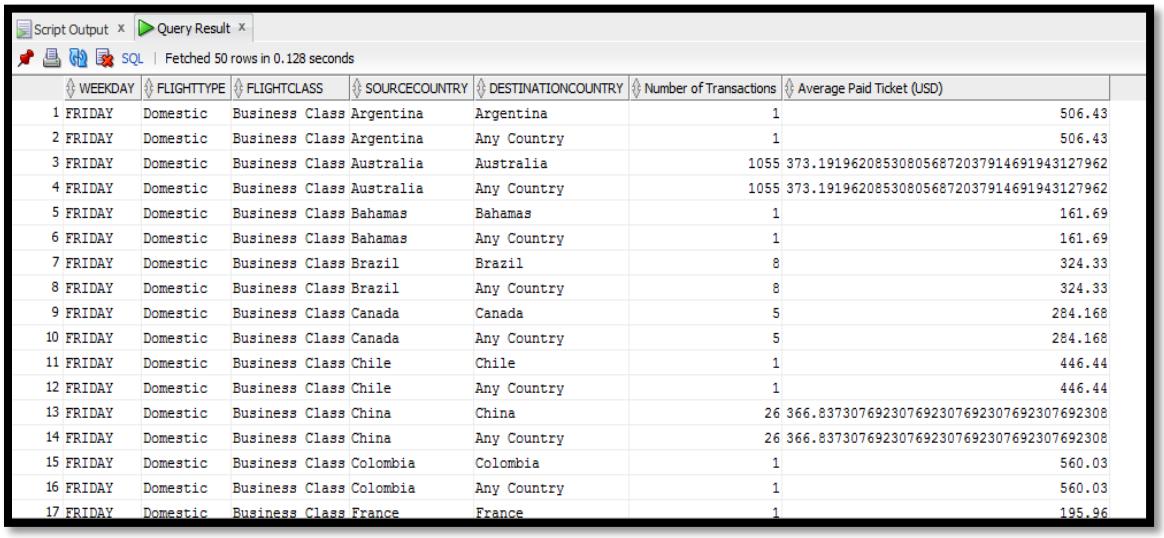
Generally, the hint of use_hash_aggregation, will produce a better execution plan and will remove the view operation from the plan as it does not save any temporary results. But in this case, there is no view operation in the original execution plan. Therefore, adding the hint of use hash aggregation is not making the query efficient but only adding an extra step of sort order by. Therefore, the original query is better.

3.6.Report 6:

a) SQL Query

```
SELECT t.WEEKDAY,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
(sum(f.SUMOFTOTALPAID)/ sum(f.TOTALNUMBEROFTRANSACTIONS)) AS
"Average Paid Ticket (USD)"
FROM passtransfact f, time_dim t, TRAVEL_CLASS_DIM tr, Flight_Type_dim ft,
source_dim s, destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.TRAVELID = tr.TRAVELID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
GROUP by  t.WEEKDAY,  rollup(ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC,
s.COUNTRY, d.COUNTRY)
order by t.WEEKDAY, ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC, s.COUNTRY,
d.COUNTRY;
```

b) Screenshot of the query result



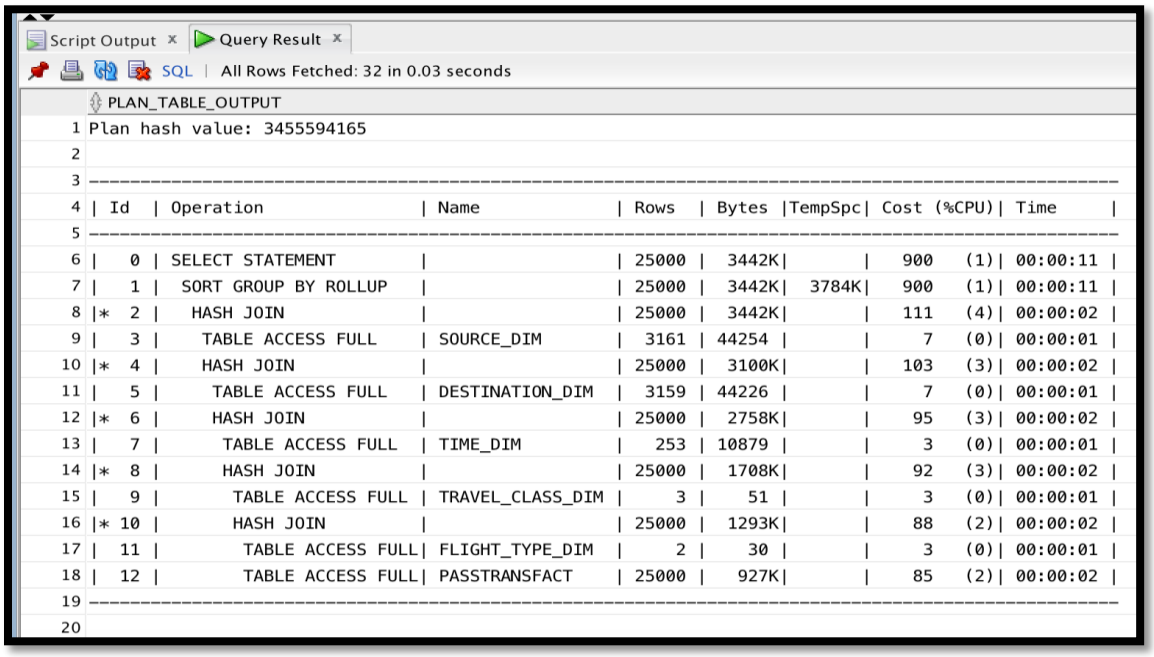
Script Output | Query Result

SQL | Fetched 50 rows in 0.128 seconds

	WEEKDAY	FLIGHTTYPE	FLIGHTCLASS	SOURCECOUNTRY	DESTINATIONCOUNTRY	Number of Transactions	Average Paid Ticket (USD)
1	FRIDAY	Domestic	Business Class	Argentina	Argentina	1	506.43
2	FRIDAY	Domestic	Business Class	Argentina	Any Country	1	506.43
3	FRIDAY	Domestic	Business Class	Australia	Australia	1055	373.191962085308056872037914691943127962
4	FRIDAY	Domestic	Business Class	Australia	Any Country	1055	373.191962085308056872037914691943127962
5	FRIDAY	Domestic	Business Class	Bahamas	Bahamas	1	161.69
6	FRIDAY	Domestic	Business Class	Bahamas	Any Country	1	161.69
7	FRIDAY	Domestic	Business Class	Brazil	Brazil	8	324.33
8	FRIDAY	Domestic	Business Class	Brazil	Any Country	8	324.33
9	FRIDAY	Domestic	Business Class	Canada	Canada	5	284.168
10	FRIDAY	Domestic	Business Class	Canada	Any Country	5	284.168
11	FRIDAY	Domestic	Business Class	Chile	Chile	1	446.44
12	FRIDAY	Domestic	Business Class	Chile	Any Country	1	446.44
13	FRIDAY	Domestic	Business Class	China	China	26	366.837307692307692307692307692308
14	FRIDAY	Domestic	Business Class	China	Any Country	26	366.837307692307692307692307692308
15	FRIDAY	Domestic	Business Class	Colombia	Colombia	1	560.03
16	FRIDAY	Domestic	Business Class	Colombia	Any Country	1	560.03
17	FRIDAY	Domestic	Business Class	France	France	1	195.96

c) Execution plan of original query

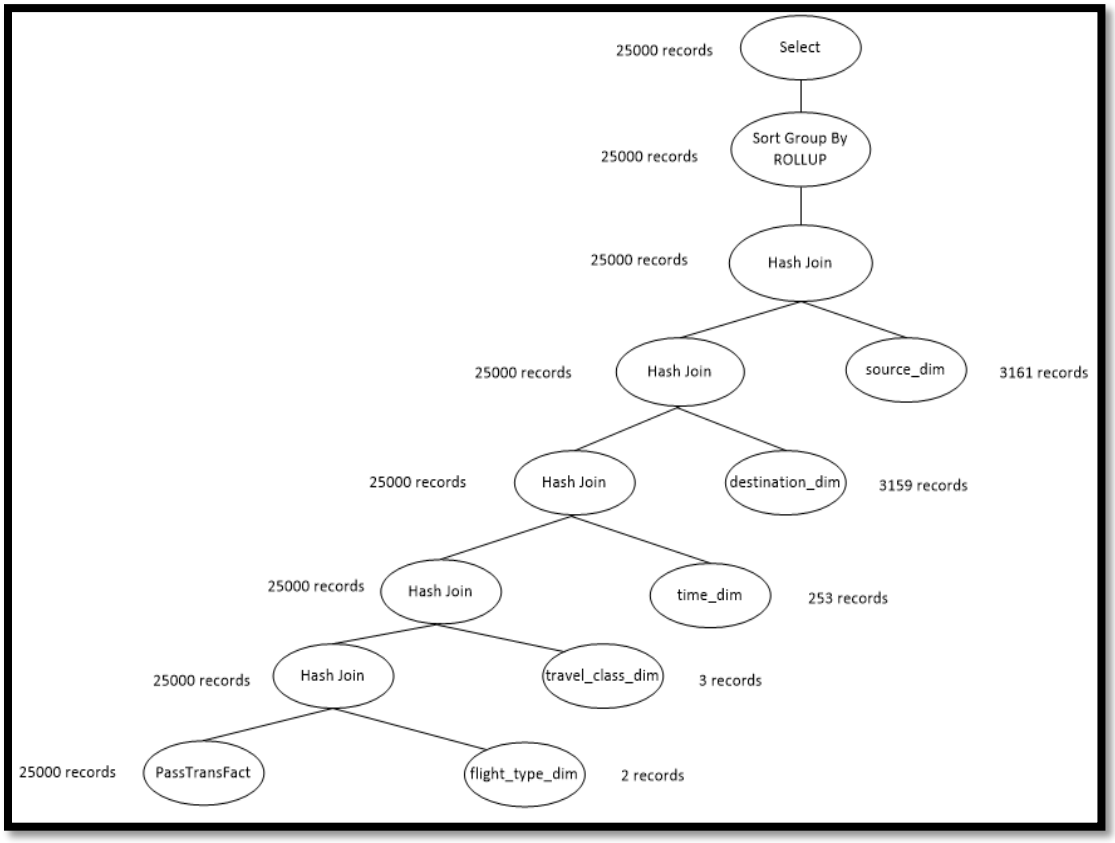
```
Explain Plan For
SELECT t.WEEKDAY,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
(sum(f.SUMOFTOTALPAID)/ sum(f.TOTALNUMBEROFTRANSACTIONS)) AS
"Average Paid Ticket (USD)"
FROM passtransfact f, time_dim t, TRAVEL_CLASS_DIM tr, Flight_Type_dim ft,
source_dim s, destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.TRAVELID = tr.TRAVELID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
GROUP by  t.WEEKDAY,  rollup(ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC,
s.COUNTRY, d.COUNTRY)
order by t.WEEKDAY, ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC, s.COUNTRY,
d.COUNTRY;

Select * From Table(dbms_xplan.display);
```



Script Output | Query Result | SQL | All Rows Fetched: 32 in 0.03 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 3455594165

-----------------------------------------------------------------------------------------------------------
| Id  | Operation               | Name              | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                   | 25000 |  3442K|       |   900   (1)| 00:00:11 |
|   1 |  SORT GROUP BY ROLLUP   |                   | 25000 |  3442K|  3784K|   900   (1)| 00:00:11 |
|*  2 |   HASH JOIN             |                   | 25000 |  3442K|       |   111   (4)| 00:00:02 |
|   3 |    TABLE ACCESS FULL    | SOURCE_DIM        |  3161 | 44254 |       |     7   (0)| 00:00:01 |
|*  4 |    HASH JOIN            |                   | 25000 |  3100K|       |   103   (3)| 00:00:02 |
|   5 |     TABLE ACCESS FULL   | DESTINATION_DIM   |  3159 | 44226 |       |     7   (0)| 00:00:01 |
|*  6 |     HASH JOIN           |                   | 25000 |  2758K|       |    95   (3)| 00:00:02 |
|   7 |      TABLE ACCESS FULL  | TIME_DIM          |   253 | 10879 |       |     3   (0)| 00:00:01 |
|*  8 |      HASH JOIN          |                   | 25000 |  1708K|       |    92   (3)| 00:00:02 |
|   9 |       TABLE ACCESS FULL | TRAVEL_CLASS_DIM  |     3 |    51 |       |     3   (0)| 00:00:01 |
|* 10 |       HASH JOIN         |                   | 25000 |  1293K|       |    88   (2)| 00:00:02 |
|  11 |        TABLE ACCESS FULL| FLIGHT_TYPE_DIM   |     2 |    30 |       |     3   (0)| 00:00:01 |
|  12 |        TABLE ACCESS FULL| PASSTRANSFACT     | 25000 |   927K|       |    85   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------------------
```

d) Query tree of the original query

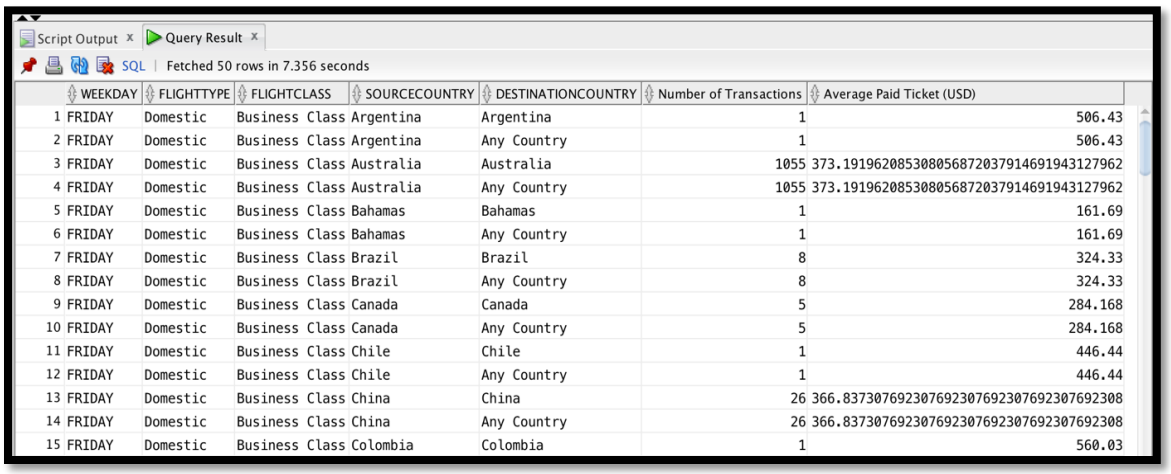


e) New SQL

```
SELECT /*+ USE_MERGE (ft tr) USE_NL (t s d f) */t.WEEKDAY,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
(sum(f.SUMOFTOTALPAID)/ sum(f.TOTALNUMBEROFTRANSACTIONS)) AS
"Average Paid Ticket (USD)"
FROM passtransfact f, time_dim t, TRAVEL_CLASS_DIM tr, Flight_Type_dim ft,
source_dim s, destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.TRAVELID = tr.TRAVELID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
```

```
GROUP by  t.WEEKDAY,  rollup(ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC,
s.COUNTRY, d.COUNTRY)
order by t.WEEKDAY, ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC, s.COUNTRY,
d.COUNTRY;
```

f) Screenshot of the new query result



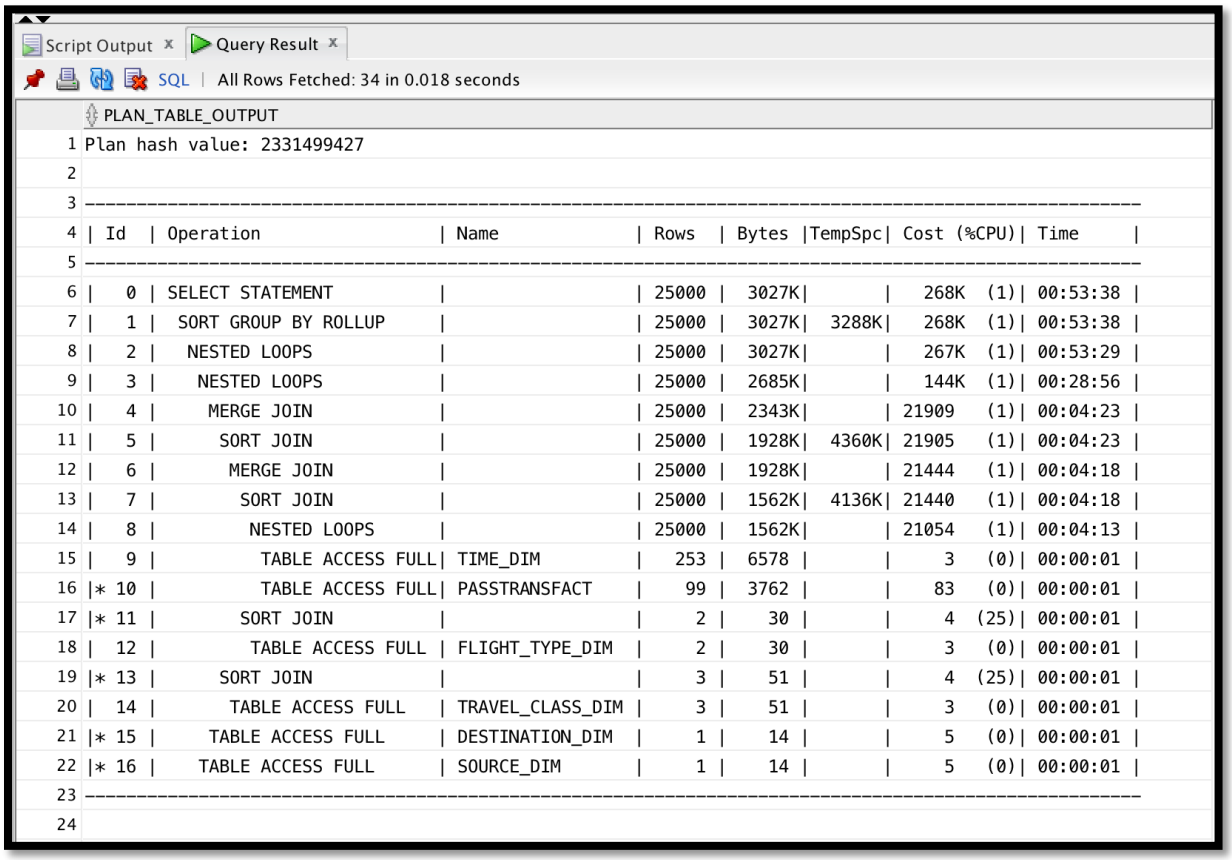
Script Output | Query Result | SQL | Fetched 50 rows in 7.356 seconds

	WEEKDAY	FLIGHTTYPE	FLIGHTCLASS	SOURCECOUNTRY	DESTINATIONCOUNTRY	Number of Transactions	Average Paid Ticket (USD)
1	FRIDAY	Domestic	Business Class	Argentina	Argentina	1	506.43
2	FRIDAY	Domestic	Business Class	Argentina	Any Country	1	506.43
3	FRIDAY	Domestic	Business Class	Australia	Australia	1055	373.19196208530805687203791469194312796 2
4	FRIDAY	Domestic	Business Class	Australia	Any Country	1055	373.19196208530805687203791469194312796 2
5	FRIDAY	Domestic	Business Class	Bahamas	Bahamas	1	161.69
6	FRIDAY	Domestic	Business Class	Bahamas	Any Country	1	161.69
7	FRIDAY	Domestic	Business Class	Brazil	Brazil	8	324.33
8	FRIDAY	Domestic	Business Class	Brazil	Any Country	8	324.33
9	FRIDAY	Domestic	Business Class	Canada	Canada	5	284.168
10	FRIDAY	Domestic	Business Class	Canada	Any Country	5	284.168
11	FRIDAY	Domestic	Business Class	Chile	Chile	1	446.44
12	FRIDAY	Domestic	Business Class	Chile	Any Country	1	446.44
13	FRIDAY	Domestic	Business Class	China	China	26	366.83730769230769230769230769230769230 8
14	FRIDAY	Domestic	Business Class	China	Any Country	26	366.83730769230769230769230769230769230 8
15	FRIDAY	Domestic	Business Class	Colombia	Colombia	1	560.03

g) Execution plan of new query

```
Explain Plan For
SELECT /*+ USE_MERGE (ft tr) USE_NL (t s d f) */t.WEEKDAY,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
decode(grouping(s.COUNTRY),1,'Any Country',s.COUNTRY) AS SourceCountry,
decode(grouping(d.COUNTRY),1,'Any Country',d.COUNTRY) AS DestinationCountry,
sum(f.TOTALNUMBEROFTRANSACTIONS) AS "Number of Transactions",
(sum(f.SUMOFTOTALPAID)/ sum(f.TOTALNUMBEROFTRANSACTIONS)) AS
"Average Paid Ticket (USD)"
FROM passtransfact f, time_dim t, TRAVEL_CLASS_DIM tr, Flight_Type_dim ft,
source_dim s, destination_dim d
WHERE f.TIMEID = t.TIMEID
and f.TRAVELID = tr.TRAVELID
and f.FLIGHTTYPEID = ft.FLIGHTTYPEID
and f.SOURCEAIRPORTID = s.SOURCE_ID
and f.DESTAIRPORTID = d.DESTINATION_ID
GROUP by  t.WEEKDAY,  rollup(ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC,
s.COUNTRY, d.COUNTRY)
order by t.WEEKDAY, ft.FLIGHTTYPEDESC,tr.CLASSTYPEDESC, s.COUNTRY,
d.COUNTRY;
```

Select * From Table(dbms_xplan.display);

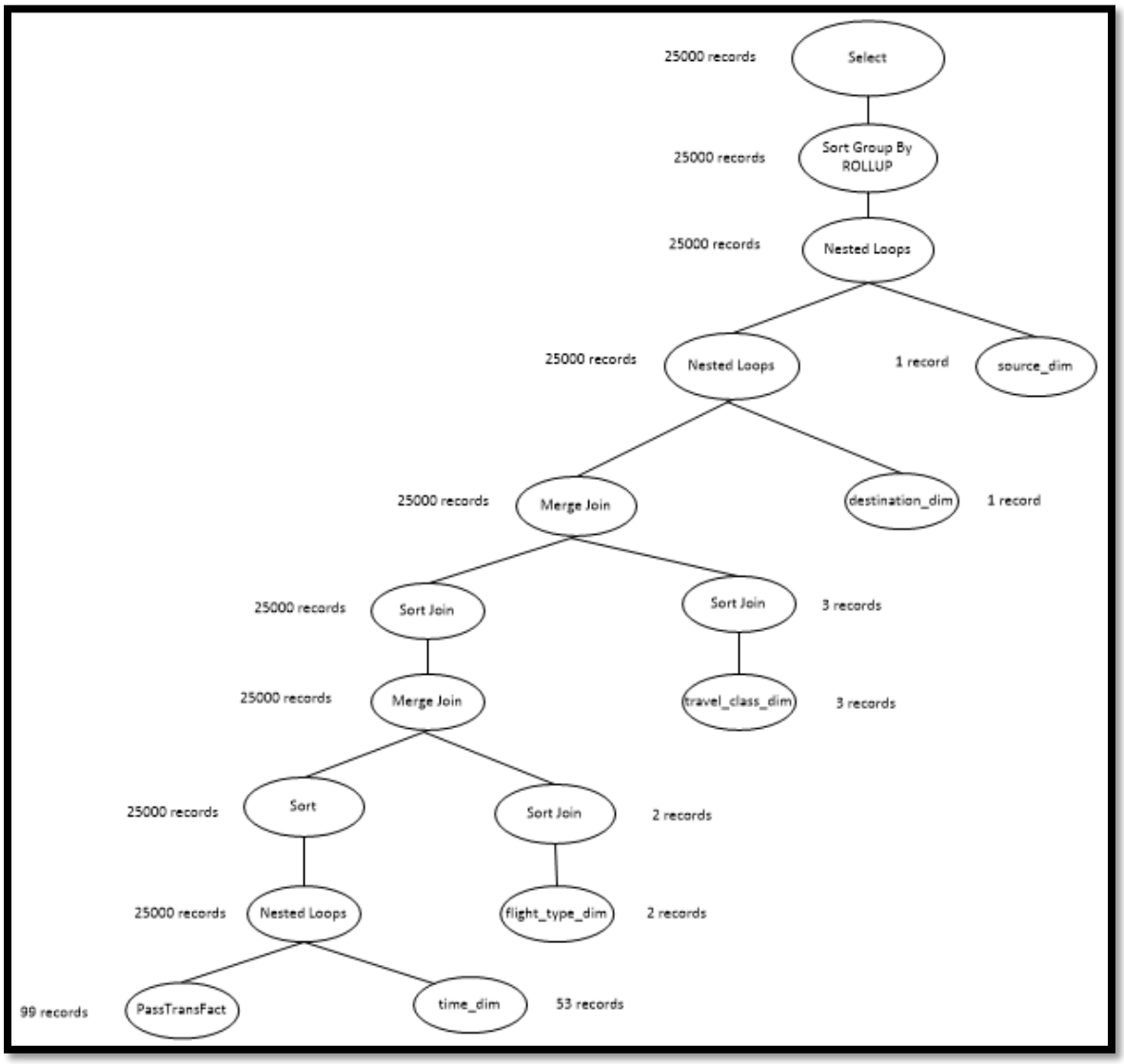


Script Output | Query Result

SQL | All Rows Fetched: 34 in 0.018 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 2331499427

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name              | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                   | 25000 |  3027K|       |   268K  (1)| 00:53:38 |
|   1 |  SORT GROUP BY ROLLUP      |                   | 25000 |  3027K|  3288K|   268K  (1)| 00:53:38 |
|   2 |   NESTED LOOPS             |                   | 25000 |  3027K|       |   267K  (1)| 00:53:29 |
|   3 |    NESTED LOOPS            |                   | 25000 |  2685K|       |   144K  (1)| 00:28:56 |
|   4 |     MERGE JOIN             |                   | 25000 |  2343K|       | 21909   (1)| 00:04:23 |
|   5 |      SORT JOIN             |                   | 25000 |  1928K|  4360K| 21905   (1)| 00:04:23 |
|   6 |       MERGE JOIN           |                   | 25000 |  1928K|       | 21444   (1)| 00:04:18 |
|   7 |        SORT JOIN           |                   | 25000 |  1562K|  4136K| 21440   (1)| 00:04:18 |
|   8 |         NESTED LOOPS       |                   | 25000 |  1562K|       | 21054   (1)| 00:04:13 |
|   9 |          TABLE ACCESS FULL| TIME_DIM          |   253 |  6578 |       |     3   (0)| 00:00:01 |
|* 10 |          TABLE ACCESS FULL| PASSTRANSFACT     |    99 |  3762 |       |    83   (0)| 00:00:01 |
|* 11 |        SORT JOIN           |                   |     2 |    30 |       |     4  (25)| 00:00:01 |
|  12 |         TABLE ACCESS FULL | FLIGHT_TYPE_DIM   |     2 |    30 |       |     3   (0)| 00:00:01 |
|* 13 |       SORT JOIN            |                   |     3 |    51 |       |     4  (25)| 00:00:01 |
|  14 |        TABLE ACCESS FULL   | TRAVEL_CLASS_DIM  |     3 |    51 |       |     3   (0)| 00:00:01 |
|* 15 |      TABLE ACCESS FULL     | DESTINATION_DIM   |     1 |    14 |       |     5   (0)| 00:00:01 |
|* 16 |     TABLE ACCESS FULL      | SOURCE_DIM        |     1 |    14 |       |     5   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

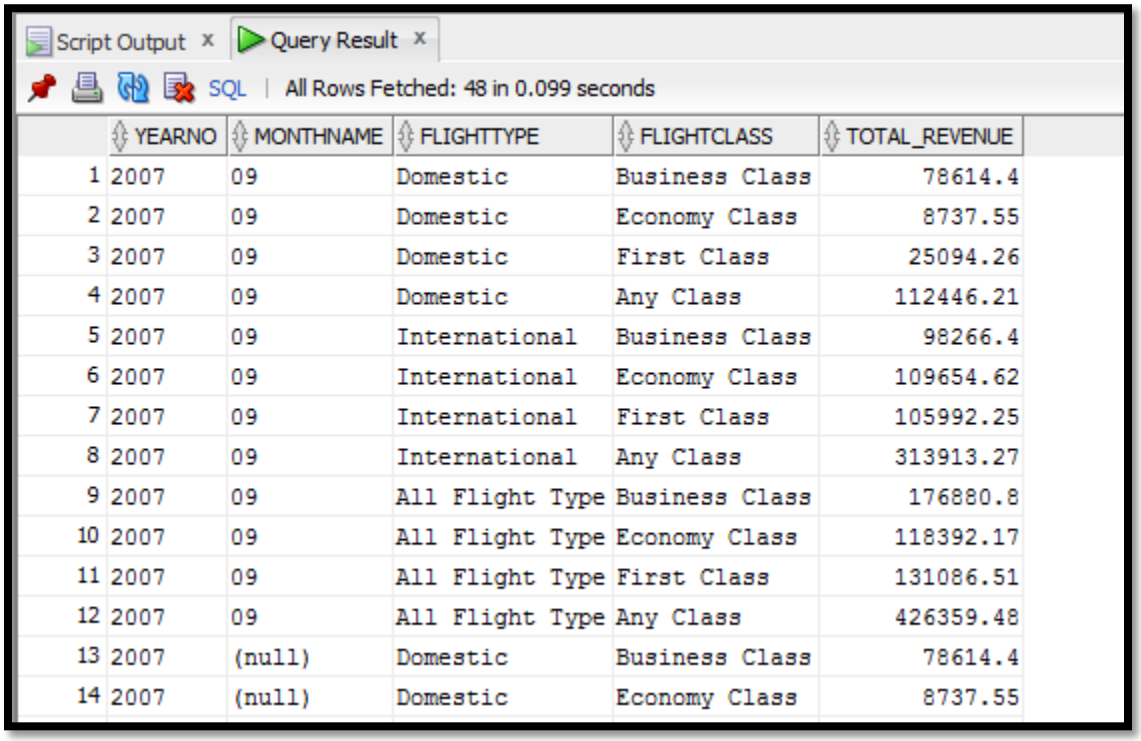
We have provided multiple hints of use_merge for tables flight_type_dim and travel_class_dim; and nested_loops for the rest four tables. The original query is optimized because the number of records in PassTransFact is very large, that is, 25000 records, hence in that case, hash join is more efficient over nested loops and merge join.

3.7.Report 7:

a) SQL Query

```
SELECT t.YEARNO, t.MONTHNAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
sum(f.SUMOFTOTALPAID) as Total_Revenue
FROM TIME_DIM t, FLIGHT_TYPE_DIM ft, TRAVEL_CLASS_DIM tr,
PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND tr.TRAVELID = f.TRAVELID
and t.YEARNO = 2007
and t.MONTHNAME = 09
GROUP BY CUBE(t.YEARNO, t.MONTHNAME, ft.FLIGHTTYPEDESC,
tr.CLASSTYPEDESC)
Order by t.YEARNO, t.MONTHNAME;
```

b) Screenshot of the query result



	YEARNO	MONTHNAME	FLIGHTTYPE	FLIGHTCLASS	TOTAL_REVENUE
1	2007	09	Domestic	Business Class	78614.4
2	2007	09	Domestic	Economy Class	8737.55
3	2007	09	Domestic	First Class	25094.26
4	2007	09	Domestic	Any Class	112446.21
5	2007	09	International	Business Class	98266.4
6	2007	09	International	Economy Class	109654.62
7	2007	09	International	First Class	105992.25
8	2007	09	International	Any Class	313913.27
9	2007	09	All Flight Type	Business Class	176880.8
10	2007	09	All Flight Type	Economy Class	118392.17
11	2007	09	All Flight Type	First Class	131086.51
12	2007	09	All Flight Type	Any Class	426359.48
13	2007	(null)	Domestic	Business Class	78614.4
14	2007	(null)	Domestic	Economy Class	8737.55

c) Execution plan of original query

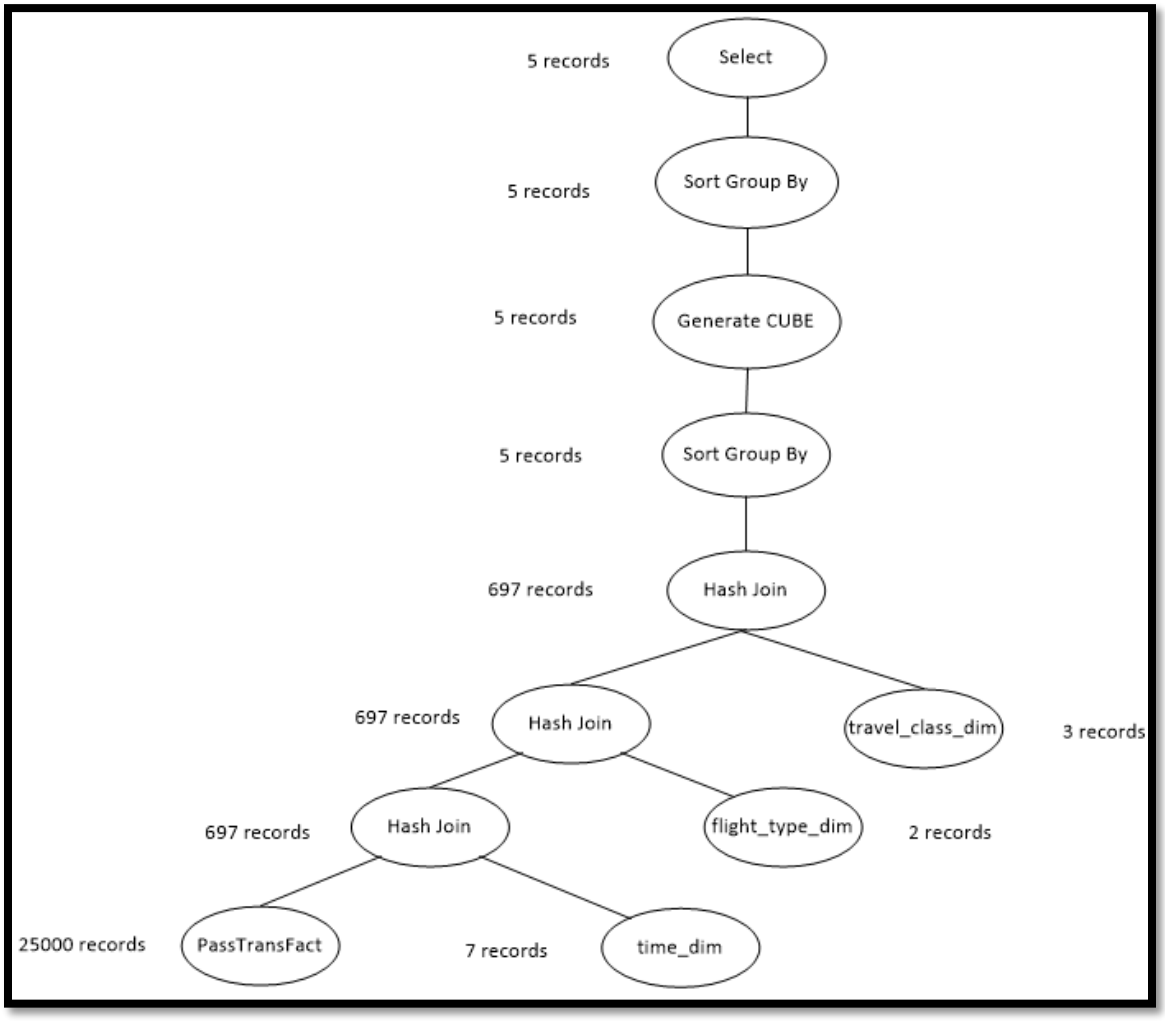
```
Explain Plan For
SELECT t.YEARNO, t.MONTHNAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
sum(f.SUMOFTOTALPAID) as Total_Revenue
FROM TIME_DIM t, FLIGHT_TYPE_DIM ft, TRAVEL_CLASS_DIM tr,
PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND tr.TRAVELID = f.TRAVELID
and t.YEARNO = 2007
and t.MONTHNAME = 09
GROUP BY CUBE(t.YEARNO, t.MONTHNAME, ft.FLIGHTTYPEDESC,
tr.CLASSTYPEDESC)
Order by t.YEARNO, t.MONTHNAME;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result | SQL | All Rows Fetched: 29 in 0.019 seconds

PLAN_TABLE_OUTPUT

```
Plan hash value: 3980956312

-----------------------------------------------------------------------------------------------
| Id  | Operation              | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                  |     5 |   445 |    96   (4)| 00:00:02 |
|   1 |  SORT GROUP BY         |                  |     5 |   445 |    96   (4)| 00:00:02 |
|   2 |   GENERATE CUBE        |                  |     5 |   445 |    96   (4)| 00:00:02 |
|   3 |    SORT GROUP BY       |                  |     5 |   445 |    96   (4)| 00:00:02 |
|*  4 |     HASH JOIN          |                  |   697 | 62033 |    95   (3)| 00:00:02 |
|   5 |      TABLE ACCESS FULL | TRAVEL_CLASS_DIM |     3 |    51 |     3   (0)| 00:00:01 |
|*  6 |      HASH JOIN         |                  |   697 | 50184 |    92   (3)| 00:00:02 |
|   7 |       TABLE ACCESS FULL| FLIGHT_TYPE_DIM  |     2 |    30 |     3   (0)| 00:00:01 |
|*  8 |       HASH JOIN        |                  |   697 | 39729 |    88   (2)| 00:00:02 |
|*  9 |        TABLE ACCESS FULL| TIME_DIM        |     7 |   210 |     3   (0)| 00:00:01 |
|  10 |        TABLE ACCESS FULL| PASSTRANSFACT   | 25000 |   659K|    85   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------

```

d) Query tree of the original query

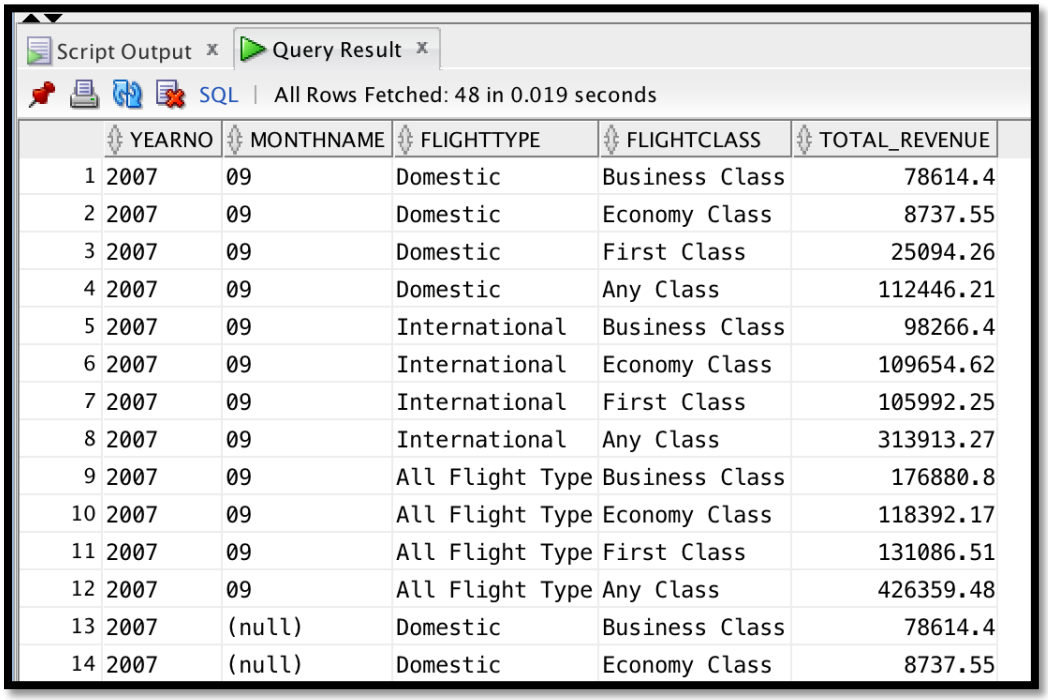


e) New SQL

```
SELECT /*+ USE_MERGE */t.YEARNO, t.MONTHNAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
sum(f.SUMOFTOTALPAID) as Total_Revenue
FROM TIME_DIM t, FLIGHT_TYPE_DIM ft, TRAVEL_CLASS_DIM tr,
PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND tr.TRAVELID = f.TRAVELID
and t.YEARNO = 2007
and t.MONTHNAME = 09
```

```
GROUP BY CUBE(t.YEARNO, t.MONTHNAME, ft.FLIGHTTYPEDESC,
tr.CLASSTYPEDESC)
Order by t.YEARNO, t.MONTHNAME;
```

f) Screenshot of the new query result



Script Output | Query Result

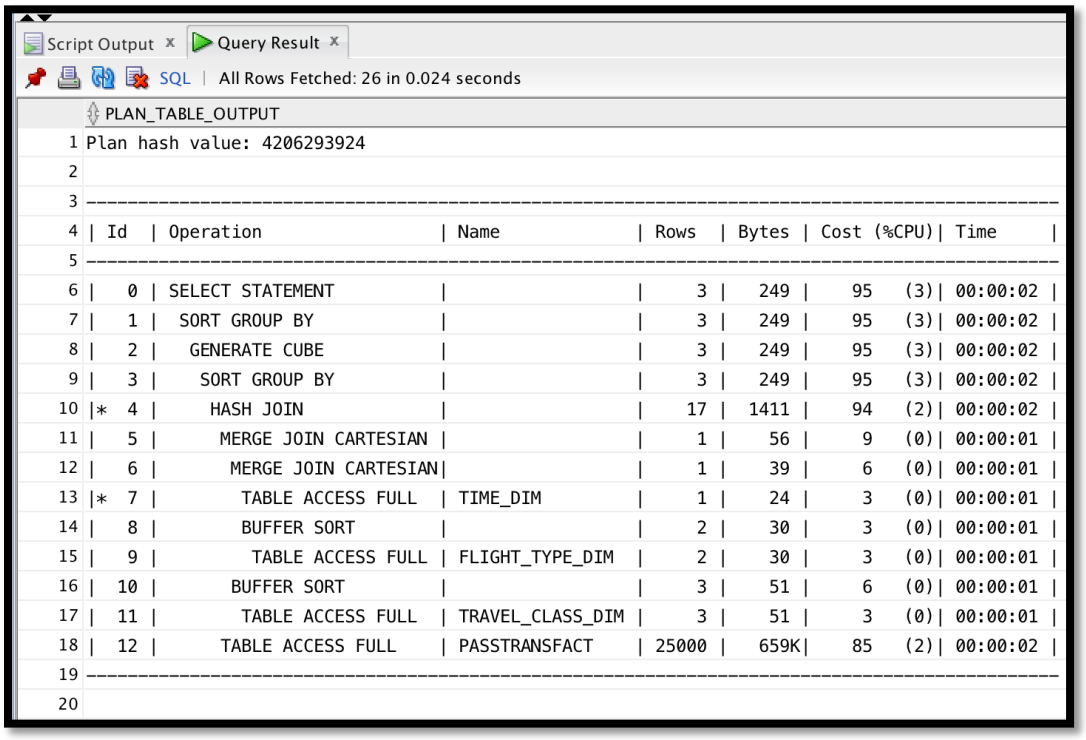
SQL | All Rows Fetched: 48 in 0.019 seconds

	YEARNO	MONTHNAME	FLIGHTTYPE	FLIGHTCLASS	TOTAL_REVENUE
1	2007	09	Domestic	Business Class	78614.4
2	2007	09	Domestic	Economy Class	8737.55
3	2007	09	Domestic	First Class	25094.26
4	2007	09	Domestic	Any Class	112446.21
5	2007	09	International	Business Class	98266.4
6	2007	09	International	Economy Class	109654.62
7	2007	09	International	First Class	105992.25
8	2007	09	International	Any Class	313913.27
9	2007	09	All Flight Type	Business Class	176880.8
10	2007	09	All Flight Type	Economy Class	118392.17
11	2007	09	All Flight Type	First Class	131086.51
12	2007	09	All Flight Type	Any Class	426359.48
13	2007	(null)	Domestic	Business Class	78614.4
14	2007	(null)	Domestic	Economy Class	8737.55

g) Execution plan of new query

```
Explain Plan For
SELECT /*+ USE_MERGE */t.YEARNO, t.MONTHNAME,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Flight Type',ft.FLIGHTTYPEDESC) AS
FlightType,
decode(grouping(tr.CLASSTYPEDESC),1,'Any Class',tr.CLASSTYPEDESC) AS
FlightClass,
sum(f.SUMOFTOTALPAID) as Total_Revenue
FROM TIME_DIM t, FLIGHT_TYPE_DIM ft, TRAVEL_CLASS_DIM tr,
PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND tr.TRAVELID = f.TRAVELID
and t.YEARNO = 2007
and t.MONTHNAME = 09
GROUP BY CUBE(t.YEARNO, t.MONTHNAME, ft.FLIGHTTYPEDESC,
tr.CLASSTYPEDESC)
Order by t.YEARNO, t.MONTHNAME;
```

```
Select * From Table(dbms_xplan.display);
```

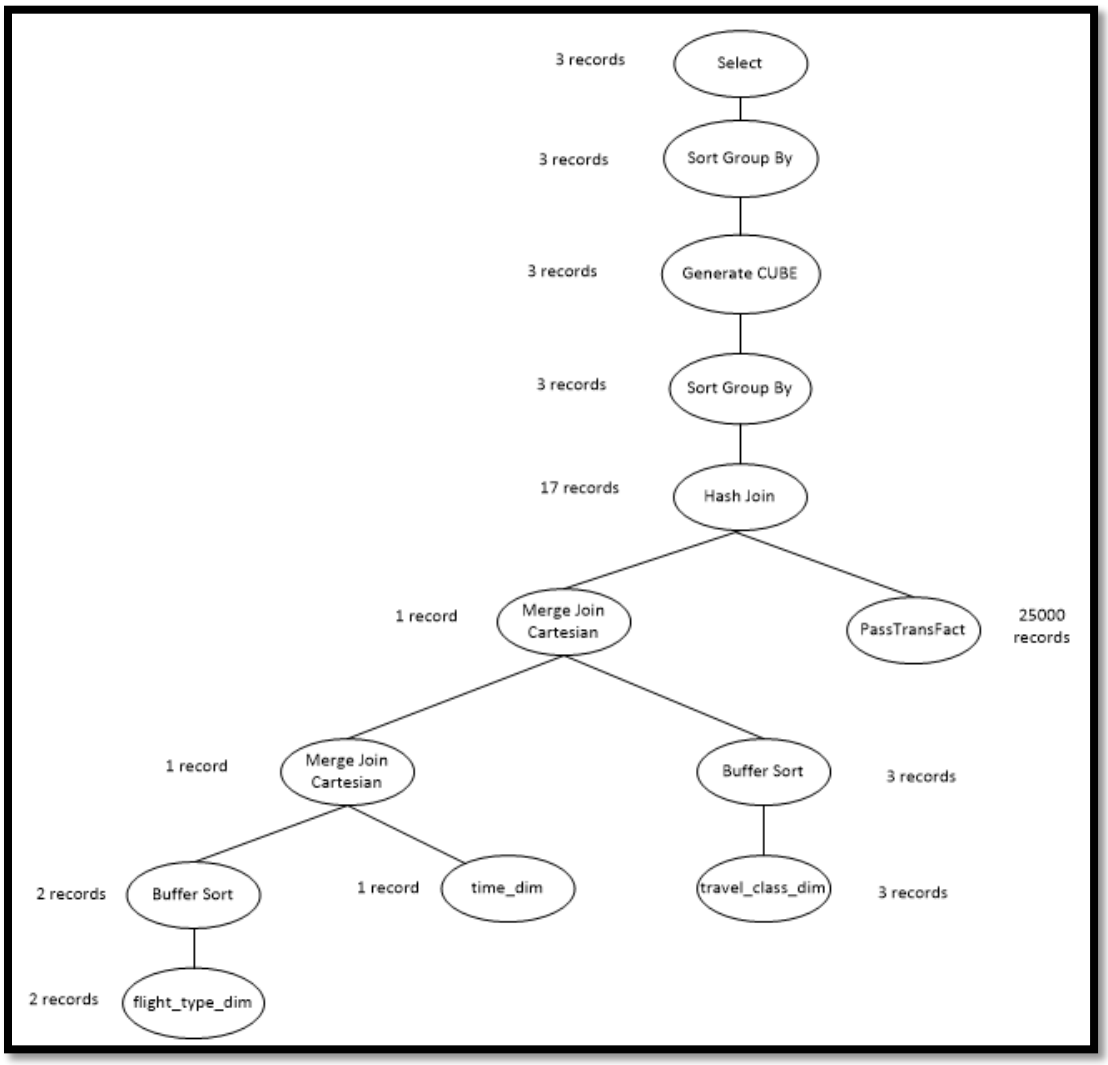


Script Output | Query Result

SQL | All Rows Fetched: 26 in 0.024 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 4206293924

-----------------------------------------------------------------------------------------------------
| Id  | Operation                | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                  |     3 |   249 |    95   (3)| 00:00:02 |
|   1 |  SORT GROUP BY           |                  |     3 |   249 |    95   (3)| 00:00:02 |
|   2 |   GENERATE CUBE          |                  |     3 |   249 |    95   (3)| 00:00:02 |
|   3 |    SORT GROUP BY         |                  |     3 |   249 |    95   (3)| 00:00:02 |
|*  4 |     HASH JOIN            |                  |    17 |  1411 |    94   (2)| 00:00:02 |
|   5 |      MERGE JOIN CARTESIAN|                  |     1 |    56 |     9   (0)| 00:00:01 |
|   6 |       MERGE JOIN CARTESIAN|                 |     1 |    39 |     6   (0)| 00:00:01 |
|*  7 |        TABLE ACCESS FULL | TIME_DIM         |     1 |    24 |     3   (0)| 00:00:01 |
|   8 |        BUFFER SORT       |                  |     2 |    30 |     3   (0)| 00:00:01 |
|   9 |         TABLE ACCESS FULL| FLIGHT_TYPE_DIM  |     2 |    30 |     3   (0)| 00:00:01 |
|  10 |       BUFFER SORT        |                  |     3 |    51 |     6   (0)| 00:00:01 |
|  11 |        TABLE ACCESS FULL | TRAVEL_CLASS_DIM |     3 |    51 |     3   (0)| 00:00:01 |
|  12 |      TABLE ACCESS FULL   | PASSTRANSFACT    | 25000 |   659K|    85   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

We have used USE_MERGE as a hint to the query execution plan, hence the operation of merge join Cartesian is used between flight_type_dim and time_dim and travel_class_dim. In this case, the number of records of the dimension table, are very less, namely 2, 1, 3 records respectively, using hash for such less number of records is very expensive and should not be preferred. Therefore, in this case, the new query is more optimized as compared to the original one.

3.8.Report 8:

a) SQL Query

```
SELECT
decode(grouping(t.YEARNO),1,'All Years',t.YEARNO) as Year,
decode(grouping(a.NAME),1,'All AirLines',a.NAME) AS AirlineName,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Types',ft.FLIGHTTYPEDESC) AS
FlightType,
sum(f.TotalTravelDistance) as Total_Distance
FROM TIME_DIM t, AIRLINE_DIM a, FLIGHT_TYPE_DIM ft, PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND a.airlineid = f.airlineid
and a.name IN ('China Eastern Airlines','IndiGo Airlines','Virgin America')
GROUP BY ROLLUP(t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC)
Order by t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC;
```

b) Screenshot of the query result

Script Output | Query Result

SQL | All Rows Fetched: 25 in 0.041 seconds

	YEAR	AIRLINENAME	FLIGHTTYPE	TOTAL_DISTANCE
1	2007	China Eastern Airlines	Domestic	3783.2
2	2007	China Eastern Airlines	International	842980.53
3	2007	China Eastern Airlines	All Types	846763.73
4	2007	IndiGo Airlines	Domestic	3175.8
5	2007	IndiGo Airlines	All Types	3175.8
6	2007	Virgin America	Domestic	3883.2
7	2007	Virgin America	All Types	3883.2
8	2007	All AirLines	All Types	853822.73
9	2008	China Eastern Airlines	Domestic	10559.2
10	2008	China Eastern Airlines	International	269343.67
11	2008	China Eastern Airlines	All Types	279902.87
12	2008	IndiGo Airlines	Domestic	3822.8
13	2008	IndiGo Airlines	All Types	3822.8
14	2008	Virgin America	Domestic	7143.01
15	2008	Virgin America	All Types	7143.01
16	2008	All AirLines	All Types	290868.68
17	2009	China Eastern Airlines	Domestic	12006.96

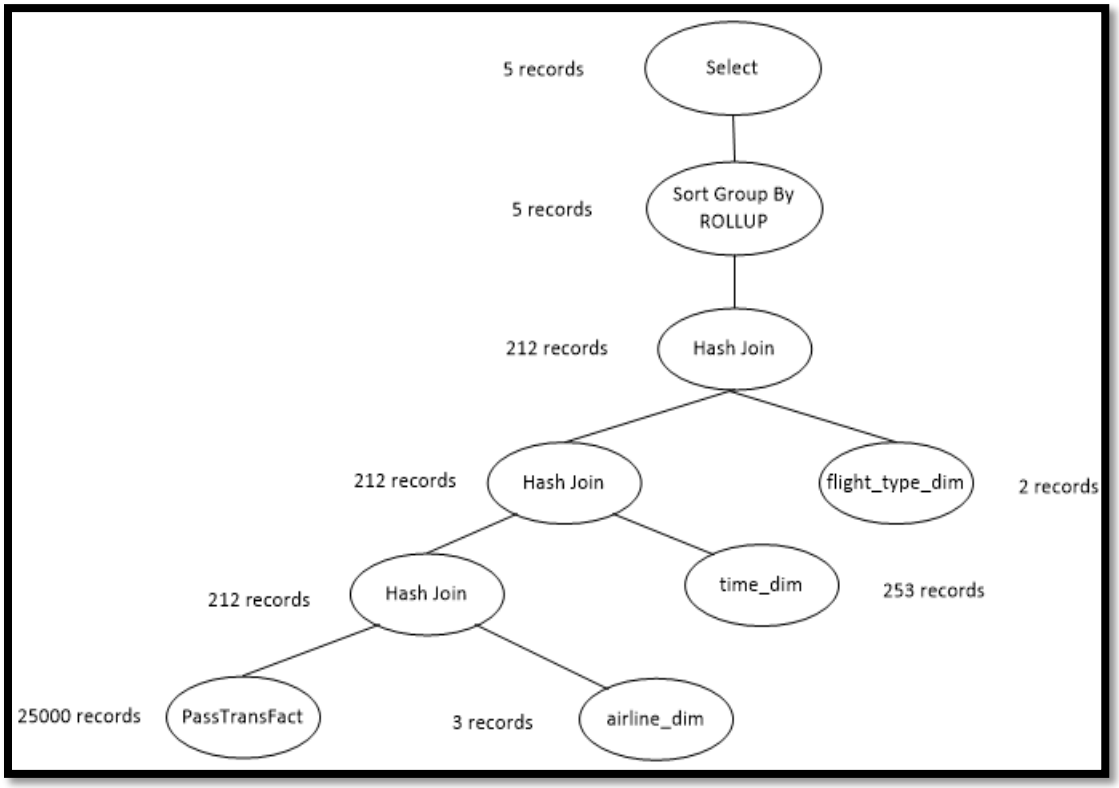
c) Execution plan of original query

```
Explain Plan For
SELECT
decode(grouping(t.YEARNO),1,'All Years',t.YEARNO) as Year,
decode(grouping(a.NAME),1,'All AirLines',a.NAME) AS AirlineName,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Types',ft.FLIGHTTYPEDESC) AS
FlightType,
sum(f.TotalTravelDistance) as Total_Distance
FROM TIME_DIM t, AIRLINE_DIM a, FLIGHT_TYPE_DIM ft, PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND a.airlineid = f.airlineid
and a.name IN ('China Eastern Airlines','IndiGo Airlines','Virgin America')
GROUP BY ROLLUP(t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC)
Order by t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result | SQL | All Rows Fetched: 28 in 0.025 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 3795183288

-----------------------------------------------------------------------------------------------
| Id  | Operation             | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                 |     5 |   475 |   107   (3)| 00:00:02 |
|   1 |  SORT GROUP BY ROLLUP |                 |     5 |   475 |   107   (3)| 00:00:02 |
|*  2 |   HASH JOIN           |                 |   212 | 20140 |   106   (2)| 00:00:02 |
|   3 |    TABLE ACCESS FULL  | FLIGHT_TYPE_DIM |     2 |    30 |     3   (0)| 00:00:01 |
|*  4 |    HASH JOIN          |                 |   212 | 16960 |   103   (2)| 00:00:02 |
|*  5 |     HASH JOIN         |                 |   212 | 11236 |    99   (2)| 00:00:02 |
|*  6 |      TABLE ACCESS FULL| AIRLINE_DIM     |     3 |    72 |    14   (0)| 00:00:01 |
|   7 |      TABLE ACCESS FULL| PASSTRANSFACT   | 25000 |  708K|    85   (2)| 00:00:02 |
|   8 |     TABLE ACCESS FULL | TIME_DIM        |   253 |  6831 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

```

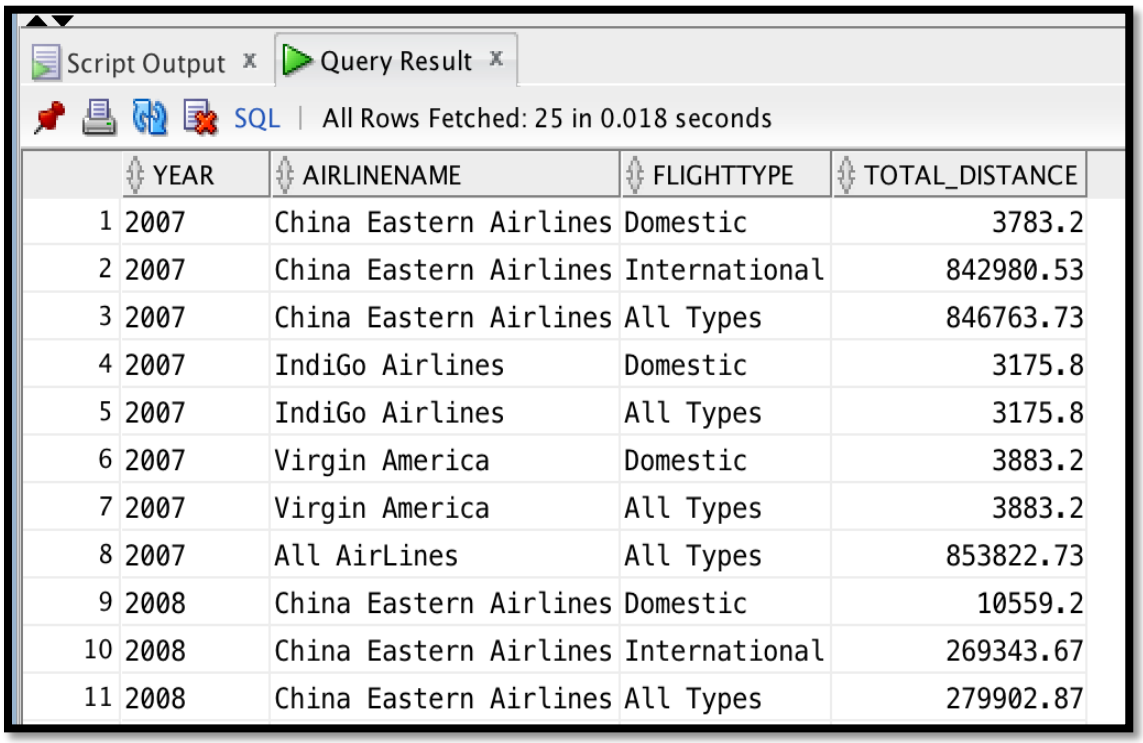
d) Query tree of the original query



e) New SQL

```
SELECT /*+ORDERED USE_MERGE (ft a) */
decode(grouping(t.YEARNO),1,'All Years',t.YEARNO) as Year,
decode(grouping(a.NAME),1,'All AirLines',a.NAME) AS AirlineName,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Types',ft.FLIGHTTYPEDESC) AS
FlightType,
sum(f.TotalTravelDistance) as Total_Distance
FROM AIRLINE_DIM a, FLIGHT_TYPE_DIM ft, TIME_DIM t, PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND a.airlineid = f.airlineid
and a.name IN ('China Eastern Airlines','IndiGo Airlines','Virgin America')
GROUP BY ROLLUP(t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC)
Order by t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC;
```

f) Screenshot of the new query result



	YEAR	AIRLINENAME	FLIGHTTYPE	TOTAL_DISTANCE
1	2007	China Eastern Airlines	Domestic	3783.2
2	2007	China Eastern Airlines	International	842980.53
3	2007	China Eastern Airlines	All Types	846763.73
4	2007	IndiGo Airlines	Domestic	3175.8
5	2007	IndiGo Airlines	All Types	3175.8
6	2007	Virgin America	Domestic	3883.2
7	2007	Virgin America	All Types	3883.2
8	2007	All AirLines	All Types	853822.73
9	2008	China Eastern Airlines	Domestic	10559.2
10	2008	China Eastern Airlines	International	269343.67
11	2008	China Eastern Airlines	All Types	279902.87

g) Execution plan of new query

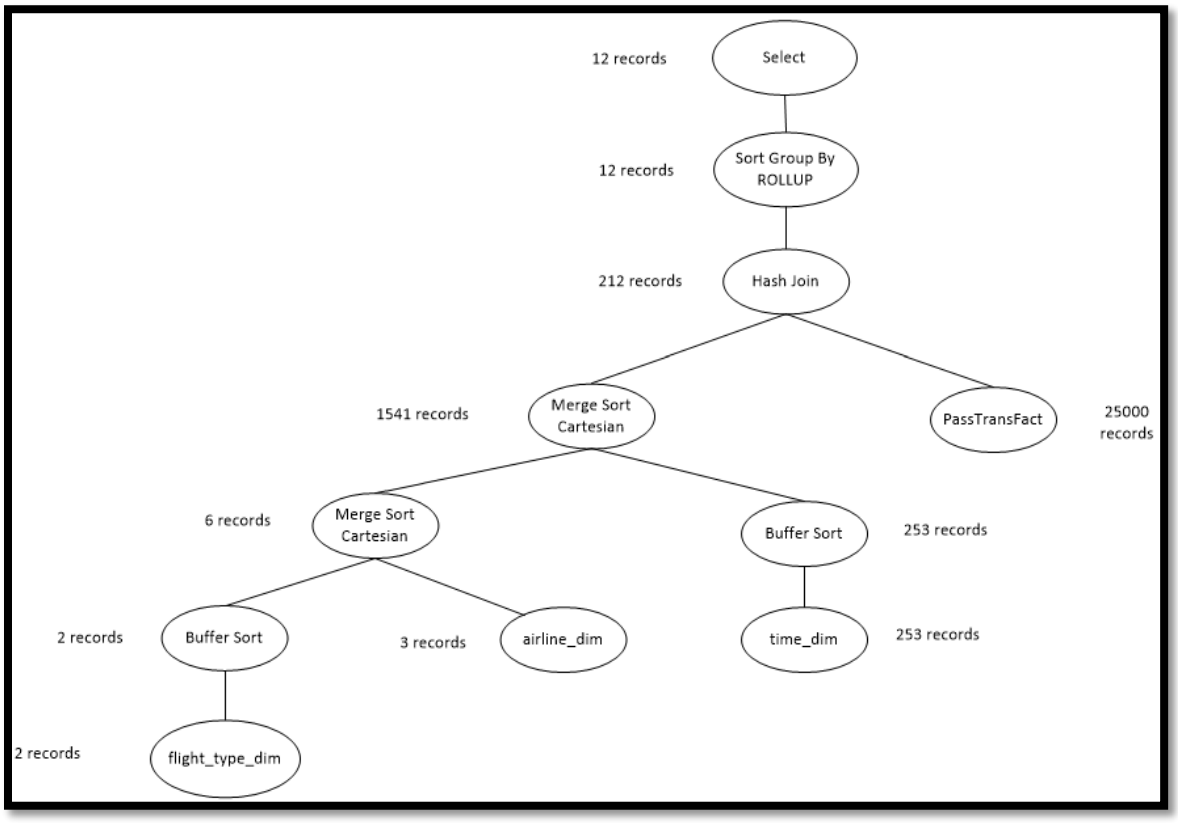
```
Explain Plan For
SELECT /*+ORDERED USE_MERGE (ft a) */
decode(grouping(t.YEARNO),1,'All Years',t.YEARNO) as Year,
decode(grouping(a.NAME),1,'All AirLines',a.NAME) AS AirlineName,
decode(grouping(ft.FLIGHTTYPEDESC),1,'All Types',ft.FLIGHTTYPEDESC) AS
FlightType,
sum(f.TotalTravelDistance) as Total_Distance
FROM AIRLINE_DIM a, FLIGHT_TYPE_DIM ft, TIME_DIM t, PASSTRANSFACT f
WHERE f.TIMEID = t.TIMEID
AND ft.FLIGHTTYPEID = f.FLIGHTTYPEID
AND a.airlineid = f.airlineid
and a.name IN ('China Eastern Airlines','IndiGo Airlines','Virgin America')
GROUP BY ROLLUP(t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC)
Order by t.YEARNO, a.NAME, ft.FLIGHTTYPEDESC;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 25 in 0.02 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 407828151

-----------------------------------------------------------------------------------------------------
| Id  | Operation                | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                  |    12 |  1068 |   116   (2)| 00:00:02 |
|   1 |  SORT GROUP BY ROLLUP    |                  |    12 |  1068 |   116   (2)| 00:00:02 |
|*  2 |   HASH JOIN              |                  |   212 | 18868 |   115   (1)| 00:00:02 |
|   3 |    MERGE JOIN CARTESIAN  |                  |  1541 | 92460 |    30   (0)| 00:00:01 |
|   4 |     MERGE JOIN CARTESIAN |                  |     6 |   234 |    20   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL   | AIRLINE_DIM      |     3 |    72 |    14   (0)| 00:00:01 |
|   6 |      BUFFER SORT         |                  |     2 |    30 |     6   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL  | FLIGHT_TYPE_DIM  |     2 |    30 |     2   (0)| 00:00:01 |
|   8 |     BUFFER SORT          |                  |   253 |  5313 |    28   (0)| 00:00:01 |
|   9 |      TABLE ACCESS FULL   | TIME_DIM         |   253 |  5313 |     2   (0)| 00:00:01 |
|  10 |    TABLE ACCESS FULL     | PASSTRANSFACT    | 25000 |  708K |    85   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

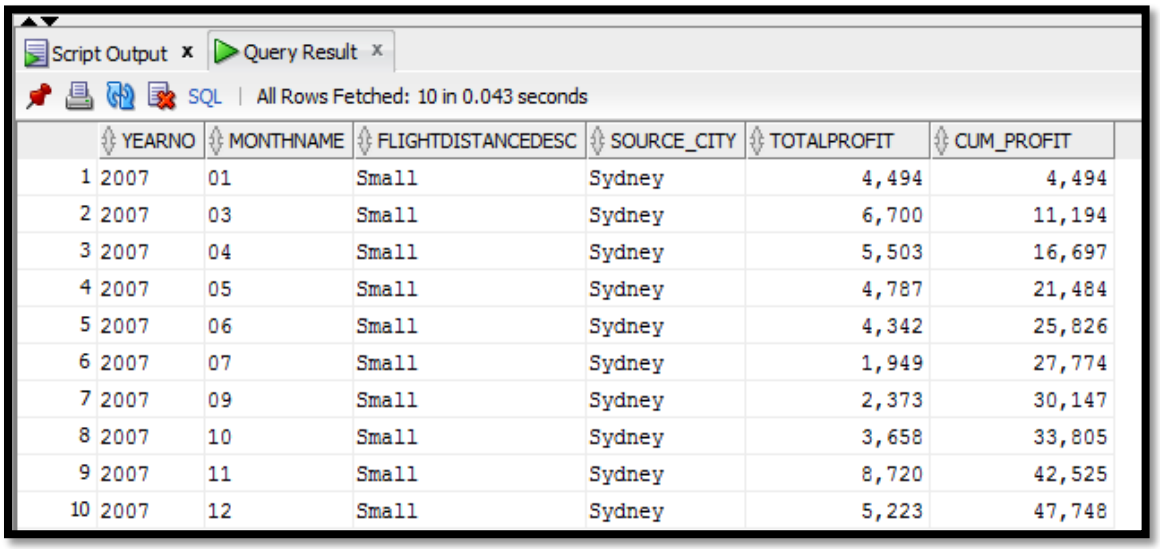
We have provided multiple hints to the query execution plan to follow merge join on flight_type_dim and airline_dim because the number of records in both tables are 2 and 3 records respectively. The hint of ordered is provided to the plan, to avoid the baggage of the number of records (25,000 records) of fact table, namely PassTransfact to be carried till the top. In our case, the original query is more efficient because the number of records are reduced to 212 records through a hash join on PassTransFact table and airline_dim table. But in a situation, where the number of records in fact table are in millions, the second query execution plan should be preferred.

3.9.Report 9:

a) SQL Query

```
SELECT t.yearNo, t.monthname, f.flightdistancedesc, s.city as source_city,
TO_CHAR (SUM(p.TotalProfit), '9,999,999,999') AS TotalProfit,
TO_CHAR (SUM(SUM(p.TotalProfit)) OVER
(ORDER BY t.yearNo, t.monthname, f.flightdistancedesc, s.city ROWS UNBOUNDED
PRECEDING),
'9,999,999,999') AS CUM_profit
FROM time_dim t, passtransfact p, flight_distance_dim f, source_dim s
WHERE t.TIMEID = p.TIMEID
AND s.SOURCE_ID = p.SOURCEAIRPORTID
and f.FLIGHTDISTANCEID = p.FLIGHTDISTANCEID
AND t.YEARNO = 2007
AND f.FLIGHTDISTANCEDESC = 'Small'
and s.CITY = 'Sydney'
GROUP BY t.yearNo, t.monthname, f.flightdistancedesc, s.city;
```

b) Screenshot of the query result



Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.043 seconds

	YEARNO	MONTHNAME	FLIGHTDISTANCEDESC	SOURCE_CITY	TOTALPROFIT	CUM_PROFIT
1	2007	01	Small	Sydney	4,494	4,494
2	2007	03	Small	Sydney	6,700	11,194
3	2007	04	Small	Sydney	5,503	16,697
4	2007	05	Small	Sydney	4,787	21,484
5	2007	06	Small	Sydney	4,342	25,826
6	2007	07	Small	Sydney	1,949	27,774
7	2007	09	Small	Sydney	2,373	30,147
8	2007	10	Small	Sydney	3,658	33,805
9	2007	11	Small	Sydney	8,720	42,525
10	2007	12	Small	Sydney	5,223	47,748

c) Execution plan of original query

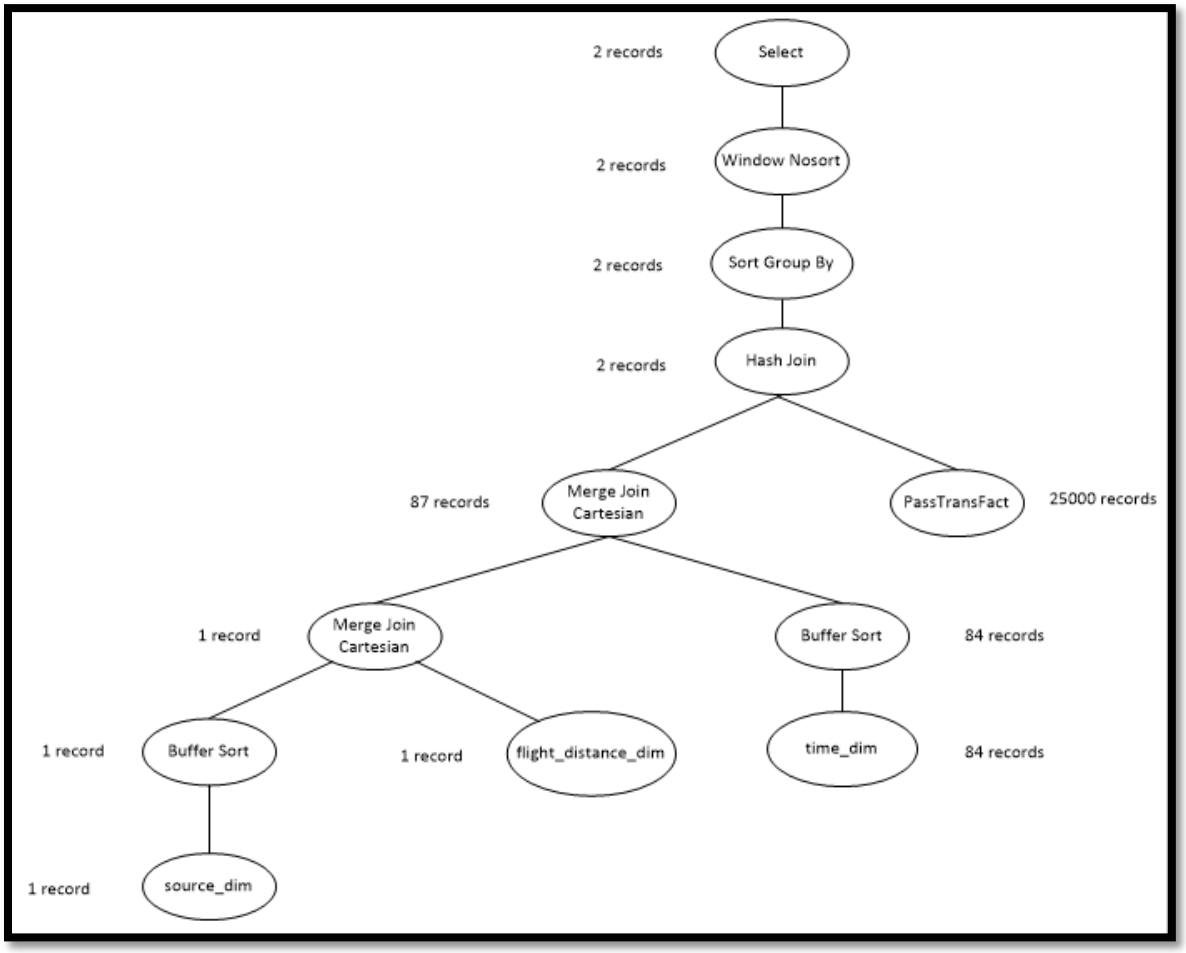
```
Explain Plan For
SELECT t.yearNo, t.monthname, f.flightdistancedesc, s.city as source_city,
TO_CHAR (SUM(p.TotalProfit), '9,999,999,999') AS TotalProfit,
TO_CHAR (SUM(SUM(p.TotalProfit)) OVER
(ORDER BY t.yearNo, t.monthname, f.flightdistancedesc, s.city ROWS UNBOUNDED
PRECEDING),
'9,999,999,999') AS CUM_profit
FROM time_dim t, passtransfact p, flight_distance_dim f, source_dim s
WHERE t.TIMEID = p.TIMEID
AND s.SOURCE_ID = p.SOURCEAIRPORTID
and f.FLIGHTDISTANCEID = p.FLIGHTDISTANCEID
AND t.YEARNO = 2007
AND f.FLIGHTDISTANCEDESC = 'Small'
and s.CITY = 'Sydney'
GROUP BY t.yearNo, t.monthname, f.flightdistancedesc, s.city;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 27 in 0.018 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 563763894

-------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                     |     2 |   154 |    99   (3)| 00:00:02 |
|   1 |  WINDOW NOSORT           |                     |     2 |   154 |    99   (3)| 00:00:02 |
|   2 |   SORT GROUP BY          |                     |     2 |   154 |    99   (3)| 00:00:02 |
|*  3 |    HASH JOIN             |                     |     2 |   154 |    98   (2)| 00:00:02 |
|   4 |     MERGE JOIN CARTESIAN |                     |    87 |  4263 |    13   (0)| 00:00:01 |
|   5 |      MERGE JOIN CARTESIAN|                     |     1 |    25 |    10   (0)| 00:00:01 |
|*  6 |       TABLE ACCESS FULL  | FLIGHT_DISTANCE_DIM |     1 |    11 |     3   (0)| 00:00:01 |
|   7 |       BUFFER SORT        |                     |     1 |    14 |     7   (0)| 00:00:01 |
|*  8 |        TABLE ACCESS FULL | SOURCE_DIM          |     1 |    14 |     7   (0)| 00:00:01 |
|   9 |      BUFFER SORT         |                     |    84 |  2016 |     6   (0)| 00:00:01 |
|* 10 |       TABLE ACCESS FULL  | TIME_DIM            |    84 |  2016 |     3   (0)| 00:00:01 |
|  11 |     TABLE ACCESS FULL    | PASSTRANSFACT       | 25000 |  683K|    85   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------------
```

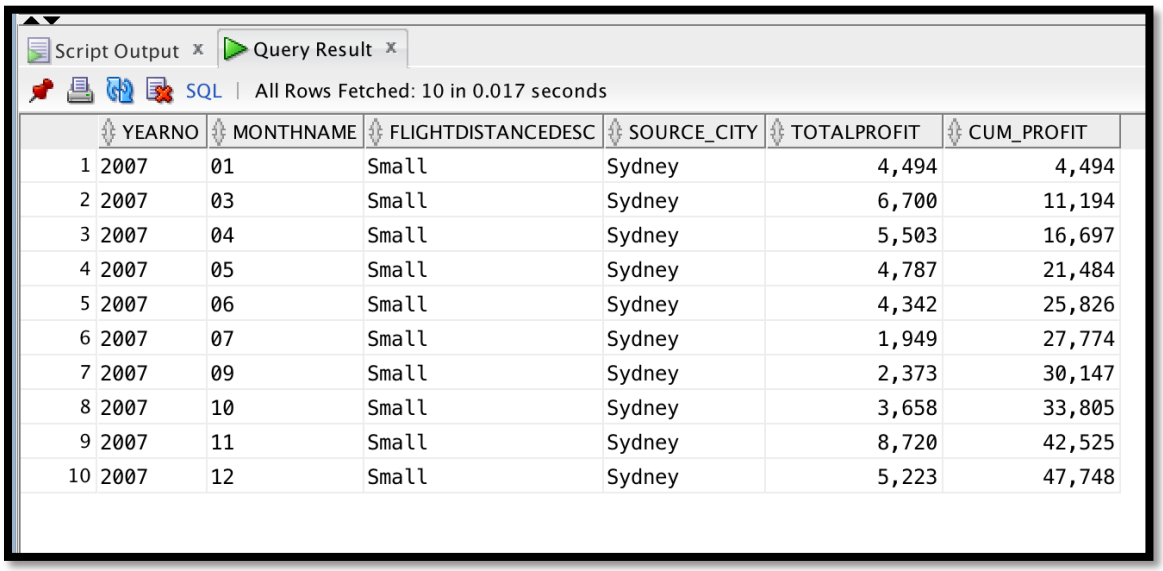
d) Query tree of the original query



e) New SQL

```
SELECT /*+ INDEX (t time_dim_index)*/t.yearNo, t.monthname, f.flightdistancedesc,
s.city as source_city,
TO_CHAR (SUM(p.TotalProfit), '9,999,999,999') AS TotalProfit,
TO_CHAR (SUM(SUM(p.TotalProfit)) OVER
(ORDER BY t.yearNo, t.monthname, f.flightdistancedesc, s.city ROWS UNBOUNDED
PRECEDING),
'9,999,999,999') AS CUM_profit
FROM time_dim t, passtransfact p, flight_distance_dim f, source_dim s
WHERE t.TIMEID = p.TIMEID
AND s.SOURCE_ID = p.SOURCEAIRPORTID
and f.FLIGHTDISTANCEID = p.FLIGHTDISTANCEID
AND t.YEARNO = 2007
AND f.FLIGHTDISTANCEDESC = 'Small'
and s.CITY = 'Sydney'
GROUP BY t.yearNo, t.monthname, f.flightdistancedesc, s.city;
```

f) Screenshot of the new query result



Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.017 seconds

	YEARNO	MONTHNAME	FLIGHTDISTANCEDESC	SOURCE_CITY	TOTALPROFIT	CUM_PROFIT
1	2007	01	Small	Sydney	4,494	4,494
2	2007	03	Small	Sydney	6,700	11,194
3	2007	04	Small	Sydney	5,503	16,697
4	2007	05	Small	Sydney	4,787	21,484
5	2007	06	Small	Sydney	4,342	25,826
6	2007	07	Small	Sydney	1,949	27,774
7	2007	09	Small	Sydney	2,373	30,147
8	2007	10	Small	Sydney	3,658	33,805
9	2007	11	Small	Sydney	8,720	42,525
10	2007	12	Small	Sydney	5,223	47,748

g) Execution plan of new query

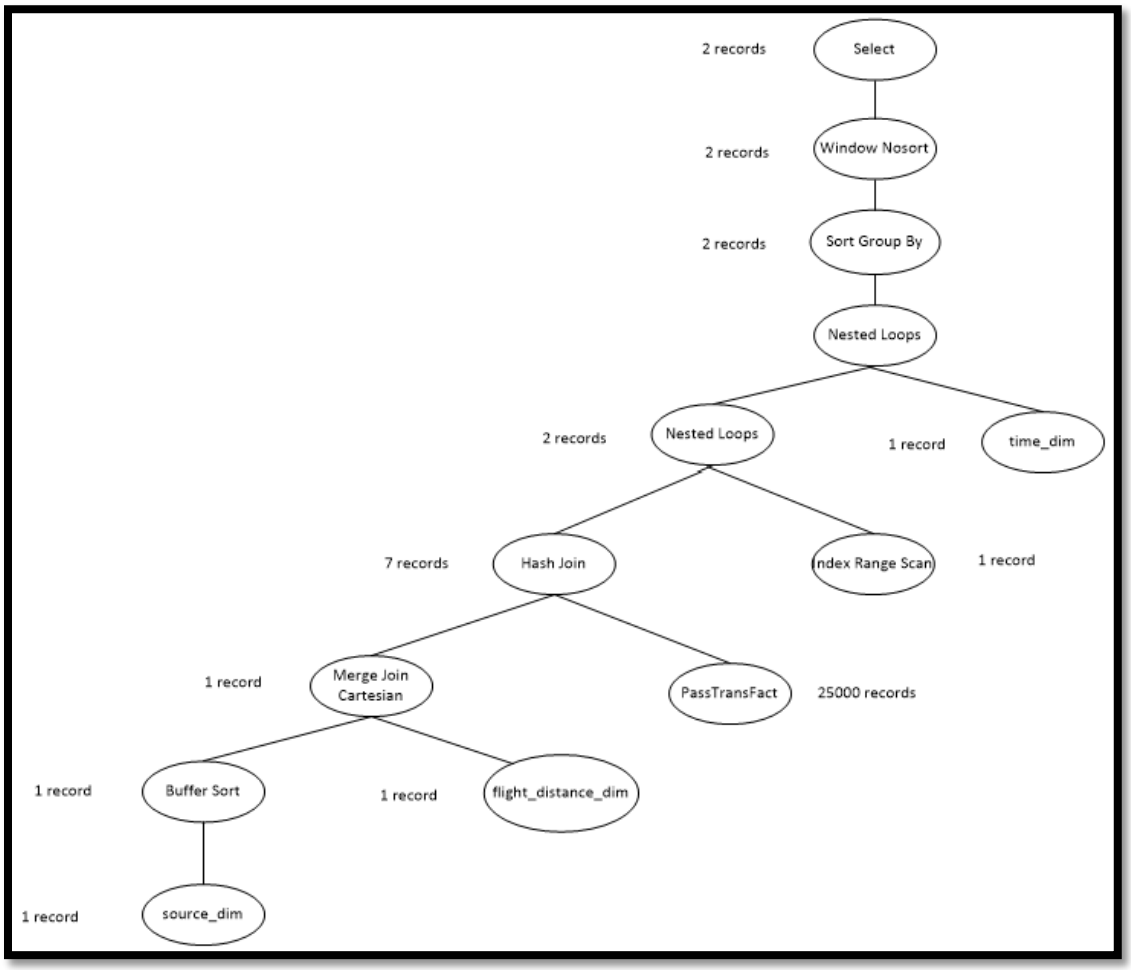
```
Create index time_dim_index
ON time_dim (timeid);

Explain Plan For
SELECT /*+ INDEX (t time_dim_index)*/t.yearNo, t.monthname, f.flightdistancedesc,
s.city as source_city,
TO_CHAR (SUM(p.TotalProfit), '9,999,999,999') AS TotalProfit,
TO_CHAR (SUM(SUM(p.TotalProfit)) OVER
(ORDER BY t.yearNo, t.monthname, f.flightdistancedesc, s.city ROWS UNBOUNDED
PRECEDING),
'9,999,999,999') AS CUM_profit
FROM time_dim t, passtransfact p, flight_distance_dim f, source_dim s
WHERE t.TIMEID = p.TIMEID
AND s.SOURCE_ID = p.SOURCEAIRPORTID
and f.FLIGHTDISTANCEID = p.FLIGHTDISTANCEID
AND t.YEARNO = 2007
AND f.FLIGHTDISTANCEDESC = 'Small'
and s.CITY = 'Sydney'
GROUP BY t.yearNo, t.monthname, f.flightdistancedesc, s.city;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 29 in 0.016 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 1232113356

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                     |     2 |   154 |   103   (2)| 00:00:02 |
|   1 |  WINDOW NOSORT                 |                     |     2 |   154 |   103   (2)| 00:00:02 |
|   2 |   SORT GROUP BY                |                     |     2 |   154 |   103   (2)| 00:00:02 |
|   3 |    NESTED LOOPS                |                     |       |       |            |          |
|   4 |     NESTED LOOPS               |                     |     2 |   154 |   102   (1)| 00:00:02 |
|*  5 |      HASH JOIN                 |                     |     7 |   371 |    95   (2)| 00:00:02 |
|   6 |       MERGE JOIN CARTESIAN     |                     |     1 |    25 |    10   (0)| 00:00:01 |
|*  7 |        TABLE ACCESS FULL       | FLIGHT_DISTANCE_DIM |     1 |    11 |     3   (0)| 00:00:01 |
|   8 |        BUFFER SORT             |                     |     1 |    14 |     7   (0)| 00:00:01 |
|*  9 |         TABLE ACCESS FULL      | SOURCE_DIM          |     1 |    14 |     7   (0)| 00:00:01 |
|  10 |       TABLE ACCESS FULL        | PASSTRANSFACT       | 25000 |  683K|    85   (2)| 00:00:02 |
|* 11 |      INDEX RANGE SCAN          | TIME_DIM_INDEX      |     1 |       |     0   (0)| 00:00:01 |
|* 12 |     TABLE ACCESS BY INDEX ROWID| TIME_DIM            |     1 |    24 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

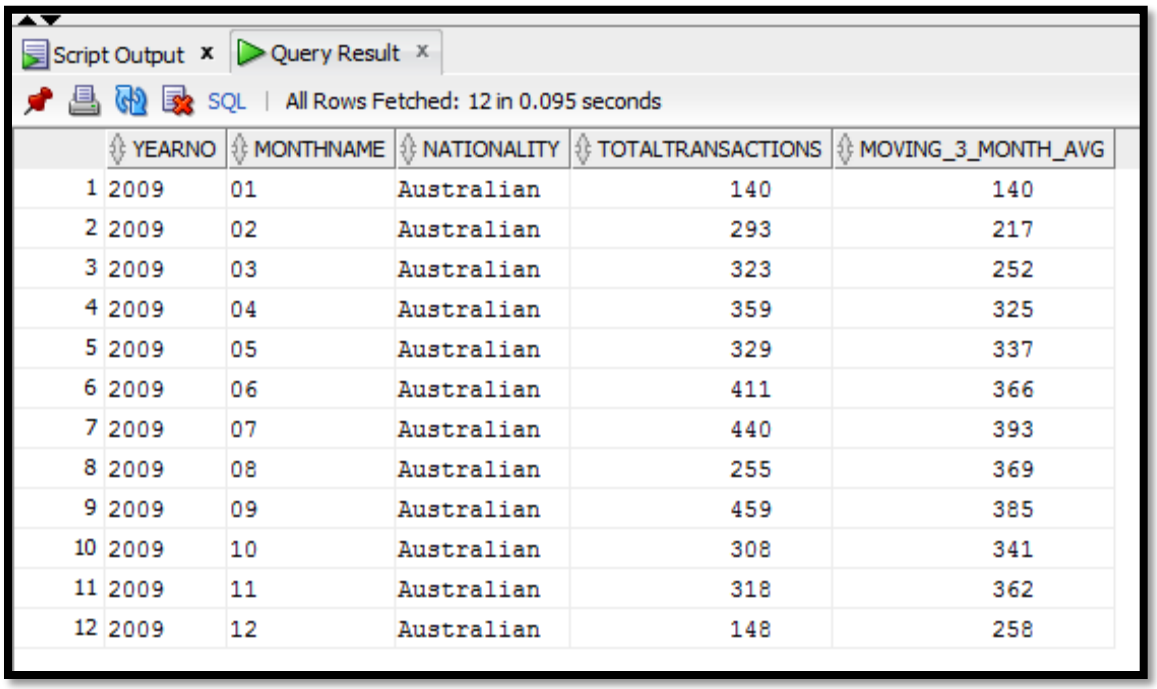
We created an index on timeid attribute from time_dim table to avoid the buffer sort step from the original execution plan. The original query is more efficient because the number of records is very less, namely 1 in source_dim and flight_distance_dim; therefore merge join Cartesian is the best operation for this query.

3.10. Report 10

a) SQL Query

```
SELECT t.YEARNO,t.MONTHNAME, n.NATIONALITY,
TO_CHAR (SUM(p.TotalNumberofTransactions), '9,999,999,999') AS TotalTransactions,
TO_CHAR (AVG(SUM(p.TotalNumberofTransactions)) OVER
(ORDER BY t.YEARNO,t.MONTHNAME, n.NATIONALITY
rows 2 preceding),
 '9,999,999,999') AS moving_3_month_avg
FROM TIME_DIM t, PASSTRANSFACT p, NATIONALITY_DIM n
WHERE t.TIMEID = p.TIMEID
AND n.NATIONALITY = p.NATIONALITY
AND t.YEARNO=2009
and n.NATIONALITY= 'Australian'
GROUP BY t.YEARNO,t.MONTHNAME, n.NATIONALITY;
```

b) Screenshot of the query result



Script Output | Query Result

SQL | All Rows Fetched: 12 in 0.095 seconds

	YEARNO	MONTHNAME	NATIONALITY	TOTALTRANSACTIONS	MOVING_3_MONTH_AVG
1	2009	01	Australian	140	140
2	2009	02	Australian	293	217
3	2009	03	Australian	323	252
4	2009	04	Australian	359	325
5	2009	05	Australian	329	337
6	2009	06	Australian	411	366
7	2009	07	Australian	440	393
8	2009	08	Australian	255	369
9	2009	09	Australian	459	385
10	2009	10	Australian	308	341
11	2009	11	Australian	318	362
12	2009	12	Australian	148	258

c) Execution plan of original query

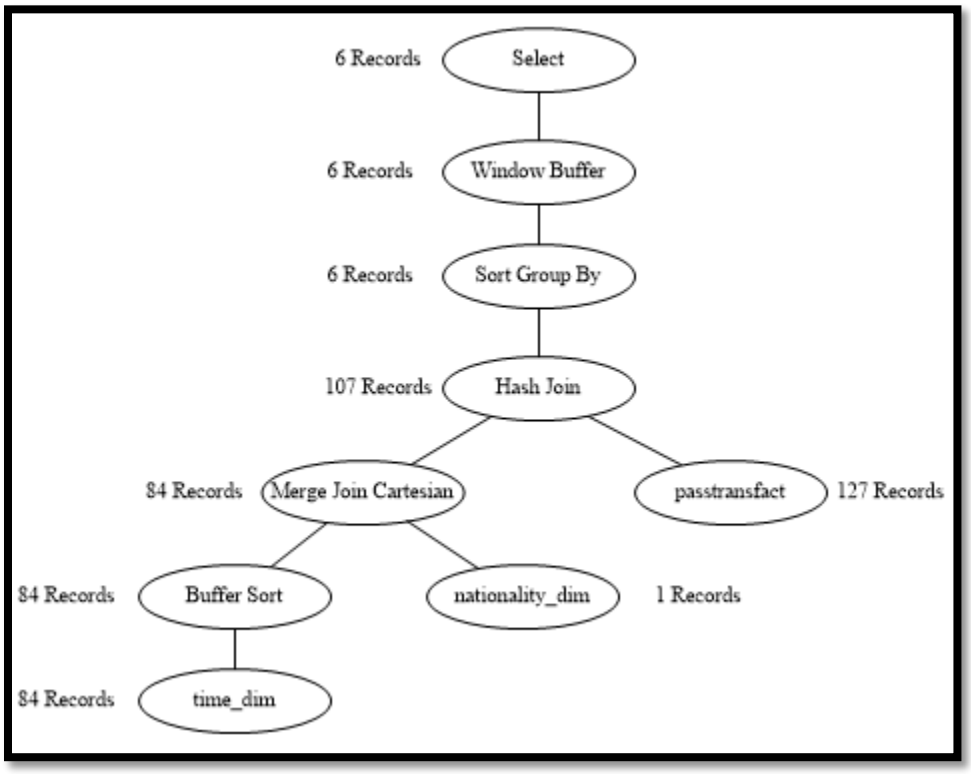
```
Explain plan for
SELECT t.YEARNO,t.MONTHNAME, n.NATIONALITY,
TO_CHAR (SUM(p.TotalNumberofTransactions), '9,999,999,999') AS TotalTransactions,
TO_CHAR (AVG(SUM(p.TotalNumberofTransactions)) OVER
(ORDER BY t.YEARNO,t.MONTHNAME, n.NATIONALITY
rows 2 preceding),
 '9,999,999,999') AS moving_3_month_avg
FROM TIME_DIM t, PASSTRANSFACT p, NATIONALITY_DIM n
WHERE t.TIMEID = p.TIMEID
AND n.NATIONALITY = p.NATIONALITY
AND t.YEARNO=2009
and n.NATIONALITY= 'Australian'
GROUP BY t.YEARNO,t.MONTHNAME, n.NATIONALITY;

Select * from table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 23 in 0.018 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 718019952

-------------------------------------------------------------------------------------------------
| Id  | Operation               | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                 |     6 |   384 |    92   (3)| 00:00:02 |
|   1 |  WINDOW BUFFER          |                 |     6 |   384 |    92   (3)| 00:00:02 |
|   2 |   SORT GROUP BY         |                 |     6 |   384 |    92   (3)| 00:00:02 |
|*  3 |    HASH JOIN            |                 |   107 |  6848 |    91   (2)| 00:00:02 |
|   4 |     MERGE JOIN CARTESIAN|                 |    84 |  2856 |     6   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL  | NATIONALITY_DIM |     1 |    10 |     3   (0)| 00:00:01 |
|   6 |      BUFFER SORT        |                 |    84 |  2016 |     3   (0)| 00:00:01 |
|*  7 |       TABLE ACCESS FULL | TIME_DIM        |    84 |  2016 |     3   (0)| 00:00:01 |
|*  8 |     TABLE ACCESS FULL   | PASSTRANSFACT   |   127 |  3810 |    84   (0)| 00:00:02 |
-------------------------------------------------------------------------------------------------

```

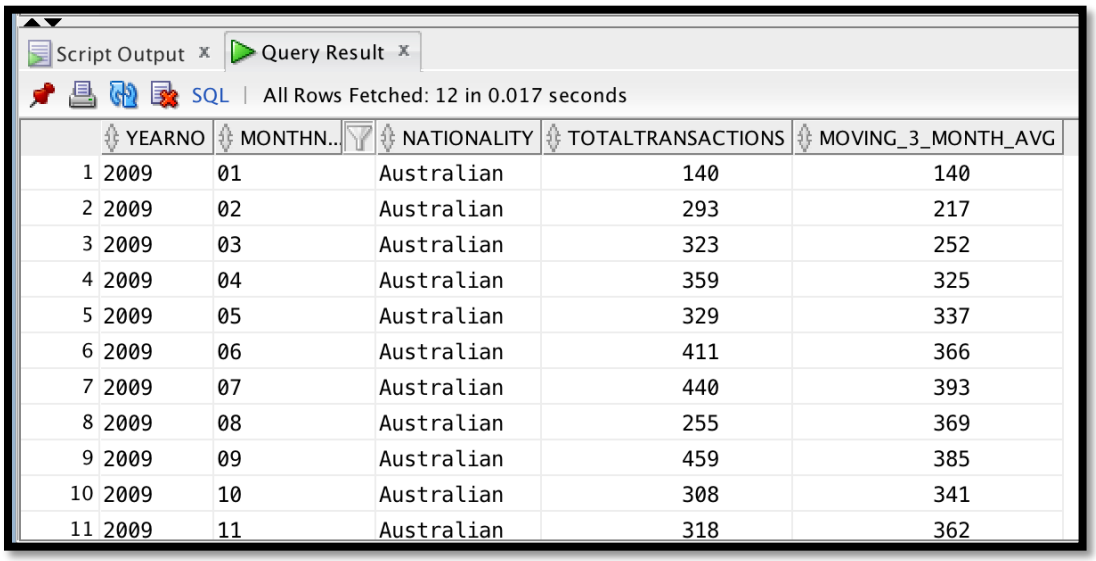
d) Query tree of the original query



e) New SQL

```
SELECT /*+ USE_NL (t n) USE_MERGE (p)*/t.YEARNO,t.MONTHNAME,
n.NATIONALITY,
TO_CHAR (SUM(p.TotalNumberofTransactions), '9,999,999,999') AS TotalTransactions,
TO_CHAR (AVG(SUM(p.TotalNumberofTransactions)) OVER
(ORDER BY t.YEARNO,t.MONTHNAME, n.NATIONALITY
rows 2 preceding),
 '9,999,999,999') AS moving_3_month_avg
FROM TIME_DIM t, PASSTRANSFACT p, NATIONALITY_DIM n
WHERE t.TIMEID = p.TIMEID
AND n.NATIONALITY = p.NATIONALITY
AND t.YEARNO=2009
and n.NATIONALITY= 'Australian'
GROUP BY t.YEARNO,t.MONTHNAME, n.NATIONALITY;
```

f) Screenshot of the new query result



Script Output | Query Result

SQL | All Rows Fetched: 12 in 0.017 seconds

	YEARNO	MONTHN..	NATIONALITY	TOTALTRANSACTIONS	MOVING_3_MONTH_AVG
1	2009	01	Australian	140	140
2	2009	02	Australian	293	217
3	2009	03	Australian	323	252
4	2009	04	Australian	359	325
5	2009	05	Australian	329	337
6	2009	06	Australian	411	366
7	2009	07	Australian	440	393
8	2009	08	Australian	255	369
9	2009	09	Australian	459	385
10	2009	10	Australian	308	341
11	2009	11	Australian	318	362

g) Execution plan of new query

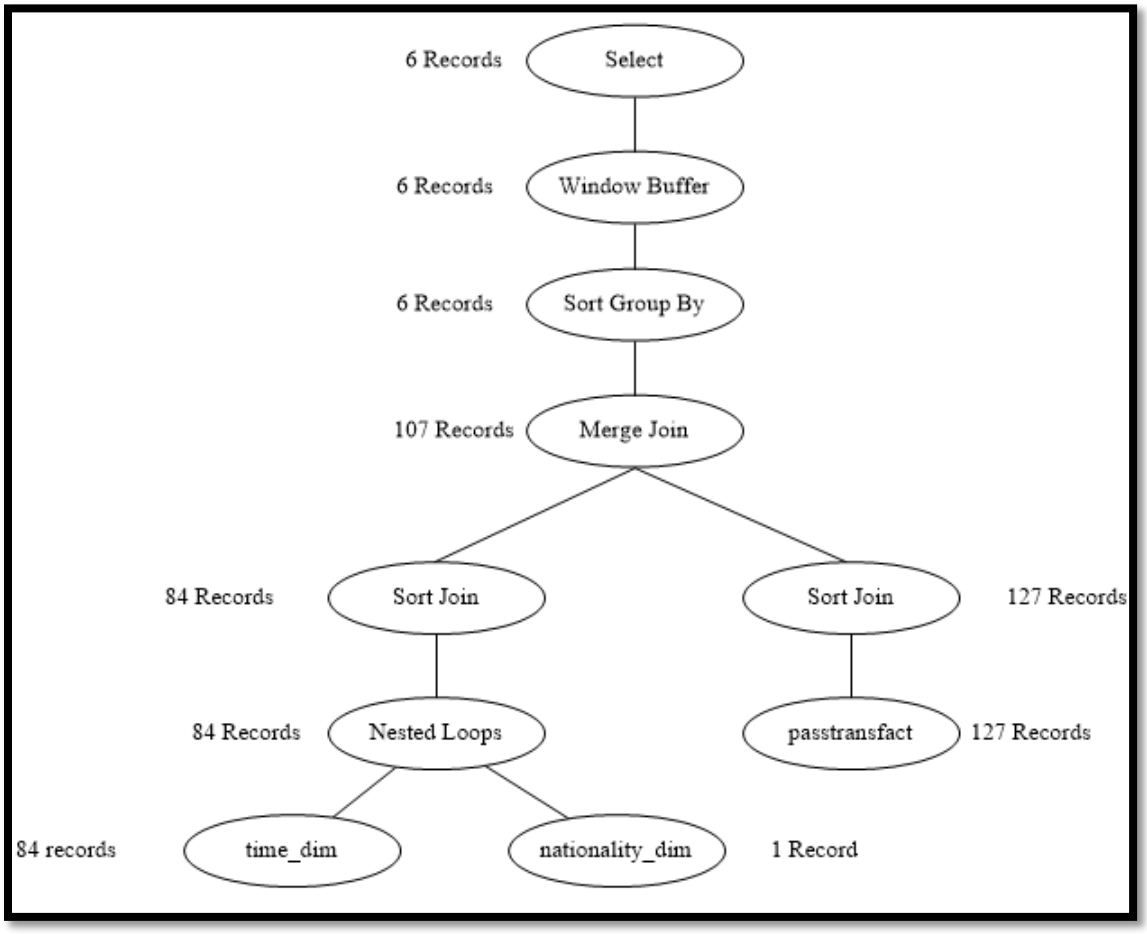
```
Explain Plan for
SELECT /*+ USE_NL (t n) USE_MERGE (p)*/t.YEARNO,t.MONTHNAME,
n.NATIONALITY,
TO_CHAR (SUM(p.TotalNumberofTransactions), '9,999,999,999') AS TotalTransactions,
TO_CHAR (AVG(SUM(p.TotalNumberofTransactions)) OVER
(ORDER BY t.YEARNO,t.MONTHNAME, n.NATIONALITY
rows 2 preceding),
 '9,999,999,999') AS moving_3_month_avg
FROM TIME_DIM t, PASSTRANSFACT p, NATIONALITY_DIM n
WHERE t.TIMEID = p.TIMEID
AND n.NATIONALITY = p.NATIONALITY
AND t.YEARNO=2009
and n.NATIONALITY= 'Australian'
GROUP BY t.YEARNO,t.MONTHNAME, n.NATIONALITY;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 25 in 0.017 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 527894407

-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                 |     6 |   384 |    93   (4)| 00:00:02 |
|   1 |  WINDOW BUFFER         |                 |     6 |   384 |    93   (4)| 00:00:02 |
|   2 |   SORT GROUP BY        |                 |     6 |   384 |    93   (4)| 00:00:02 |
|   3 |    MERGE JOIN          |                 |   107 |  6848 |    92   (3)| 00:00:02 |
|   4 |     SORT JOIN          |                 |    84 |  2856 |     7  (15)| 00:00:01 |
|   5 |      NESTED LOOPS      |                 |    84 |  2856 |     6   (0)| 00:00:01 |
|*  6 |       TABLE ACCESS FULL| NATIONALITY_DIM |     1 |    10 |     3   (0)| 00:00:01 |
|*  7 |       TABLE ACCESS FULL| TIME_DIM        |    84 |  2016 |     3   (0)| 00:00:01 |
|*  8 |     SORT JOIN          |                 |   127 |  3810 |    85   (2)| 00:00:02 |
|*  9 |      TABLE ACCESS FULL | PASSTRANSFACT   |   127 |  3810 |    84   (0)| 00:00:02 |
-------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

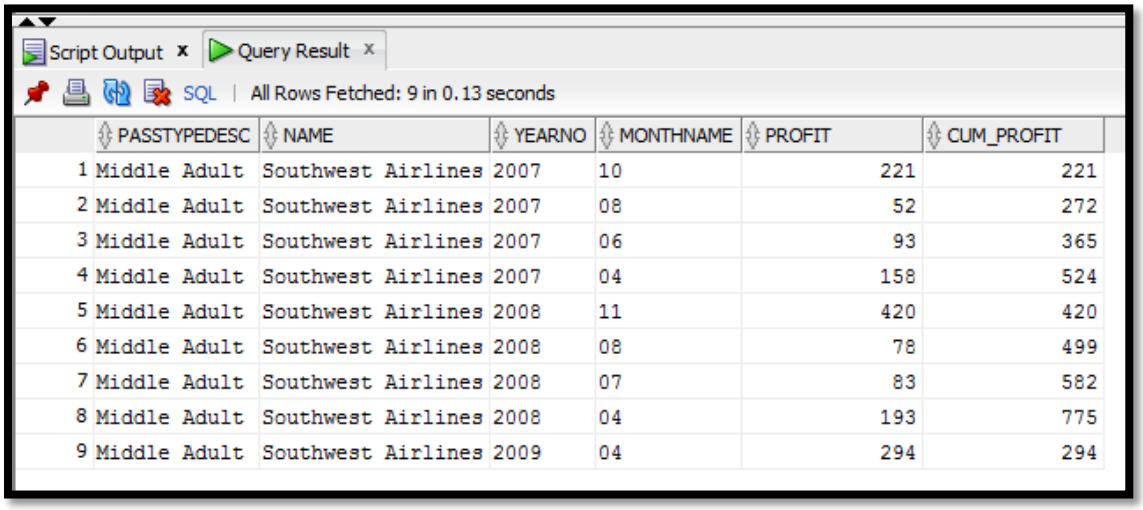
We have provided multiple hints to use nested_loops and use_merge, which have inserted additional steps of sorting in the new execution plan. The original query execution plan is better because the merge join Cartesian is performed on nationality_dim table and time_dim table and they have less number of records namely 1 and 84 records respectively.

3.11. Report 11

a) SQL Query

```
SELECT p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME,
TO_CHAR (SUM(f.TOTALPROFIT), '9,999,999,999') AS PROFIT,
TO_CHAR (SUM(SUM(f.TOTALPROFIT)) OVER
(PARTITION BY t.YEARNO ORDER BY p.PASSTYPEDESC, a.NAME, t.YEARNO,
t.MONTHNAME desc
 ROWS UNBOUNDED PRECEDING),
'9,999,999,999') AS CUM_PROFIT
FROM PASSENGER_TYPE_DIM p, PASSTRANSFACT f, TIME_DIM t, AIRLINE_DIM
a
WHERE p.passtypeid = f.passtypeid
AND t.TIMEID = f.TIMEID
AND a.AIRLINEID = f.AIRLINEID
AND a.NAME = 'Southwest Airlines'
AND p.PASSTYPEDESC = 'Middle Adult'
GROUP BY p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME;
```

b) Screenshot of the query result



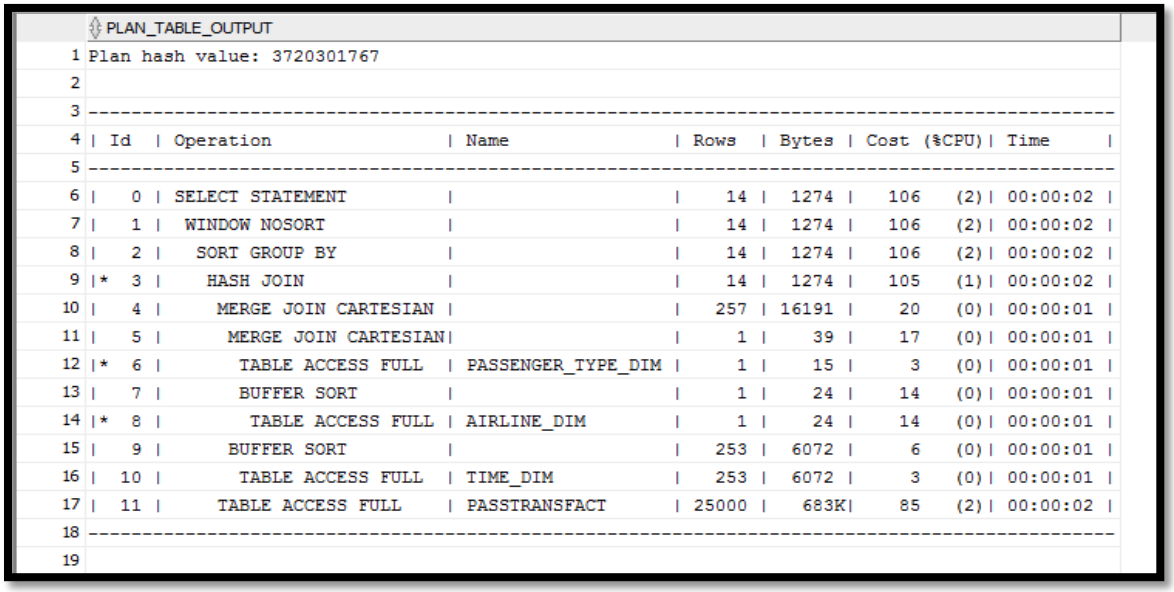
Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.13 seconds

	PASSTYPEDESC	NAME	YEARNO	MONTHNAME	PROFIT	CUM_PROFIT
1	Middle Adult	Southwest Airlines	2007	10	221	221
2	Middle Adult	Southwest Airlines	2007	08	52	272
3	Middle Adult	Southwest Airlines	2007	06	93	365
4	Middle Adult	Southwest Airlines	2007	04	158	524
5	Middle Adult	Southwest Airlines	2008	11	420	420
6	Middle Adult	Southwest Airlines	2008	08	78	499
7	Middle Adult	Southwest Airlines	2008	07	83	582
8	Middle Adult	Southwest Airlines	2008	04	193	775
9	Middle Adult	Southwest Airlines	2009	04	294	294

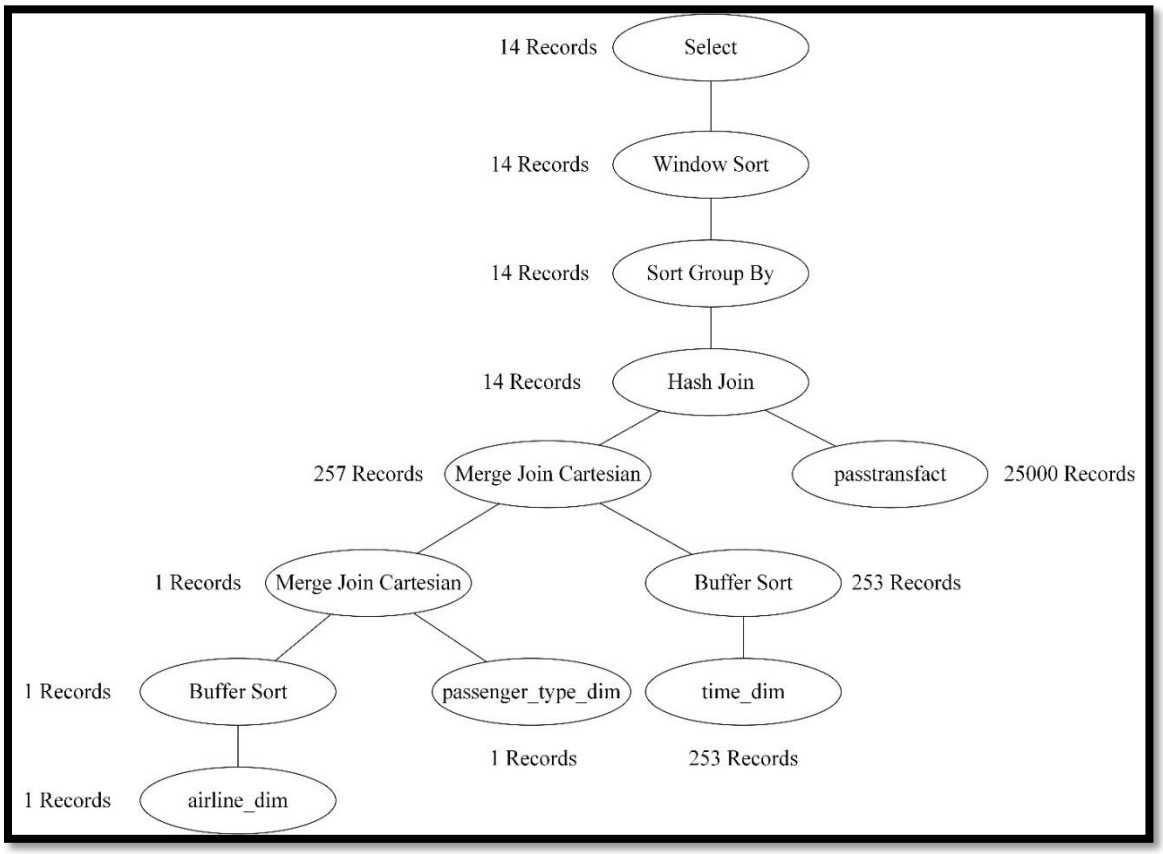
c) Execution plan of original query

```
Explain plan for
SELECT p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME,
TO_CHAR (SUM(f.TOTALPROFIT), '9,999,999,999') AS PROFIT,
TO_CHAR (SUM(SUM(f.TOTALPROFIT)) OVER
(PARTITION BY t.YEARNO ORDER BY p.PASSTYPEDESC, a.NAME, t.YEARNO,
t.MONTHNAME desc
 ROWS UNBOUNDED PRECEDING),
'9,999,999,999') AS CUM_PROFIT
FROM PASSENGER_TYPE_DIM p, PASSTRANSFACT f, TIME_DIM t, AIRLINE_DIM
a
WHERE p.passtypeid = f.passtypeid
AND t.TIMEID = f.TIMEID
AND a.AIRLINEID = f.AIRLINEID
AND a.NAME = 'Southwest Airlines'
AND p.PASSTYPEDESC = 'Middle Adult'
GROUP BY p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME;

Select * from table(dbms_xplan.display);
```



```
PLAN_TABLE_OUTPUT
Plan hash value: 3720301767

-----------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                     |    14 |  1274 |   106   (2)| 00:00:02 |
|   1 |  WINDOW NOSORT           |                     |    14 |  1274 |   106   (2)| 00:00:02 |
|   2 |   SORT GROUP BY          |                     |    14 |  1274 |   106   (2)| 00:00:02 |
|*  3 |    HASH JOIN             |                     |    14 |  1274 |   105   (1)| 00:00:02 |
|   4 |     MERGE JOIN CARTESIAN |                     |   257 | 16191 |    20   (0)| 00:00:01 |
|   5 |      MERGE JOIN CARTESIAN|                     |     1 |    39 |    17   (0)| 00:00:01 |
|*  6 |       TABLE ACCESS FULL  | PASSENGER_TYPE_DIM  |     1 |    15 |     3   (0)| 00:00:01 |
|   7 |       BUFFER SORT        |                     |     1 |    24 |    14   (0)| 00:00:01 |
|*  8 |        TABLE ACCESS FULL | AIRLINE_DIM         |     1 |    24 |    14   (0)| 00:00:01 |
|   9 |      BUFFER SORT         |                     |   253 |  6072 |     6   (0)| 00:00:01 |
|  10 |       TABLE ACCESS FULL  | TIME_DIM            |   253 |  6072 |     3   (0)| 00:00:01 |
|  11 |     TABLE ACCESS FULL    | PASSTRANSFACT       | 25000 |   683K|    85   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------------
```

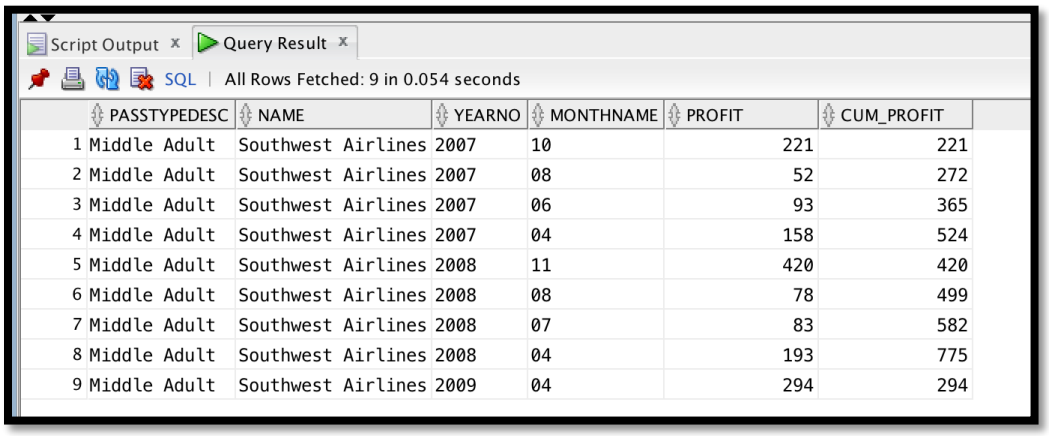
d) Query tree of the original query



e) New SQL

```
SELECT /*+ USE_NL(p a) USE_MERGE(f) */ p.PASSTYPEDESC, a.NAME, t.YEARNO,
t.MONTHNAME,
TO_CHAR (SUM(f.TOTALPROFIT), '9,999,999,999') AS PROFIT,
TO_CHAR (SUM(SUM(f.TOTALPROFIT)) OVER
(PARTITION BY t.YEARNO ORDER BY p.PASSTYPEDESC, a.NAME, t.YEARNO,
t.MONTHNAME desc
 ROWS UNBOUNDED PRECEDING),
'9,999,999,999') AS CUM_PROFIT
FROM PASSENGER_TYPE_DIM p, AIRLINE_DIM a, TIME_DIM t, PASSTRANSFACT
f
WHERE p.passtypeid = f.passtypeid
AND t.TIMEID = f.TIMEID
AND a.AIRLINEID = f.AIRLINEID
AND a.NAME = 'Southwest Airlines'
AND p.PASSTYPEDESC = 'Middle Adult'
GROUP BY p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME;
```

f) Screenshot of the new query result



Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.054 seconds

	PASSTYPEDESC	NAME	YEARNO	MONTHNAME	PROFIT	CUM_PROFIT
1	Middle Adult	Southwest Airlines	2007	10	221	221
2	Middle Adult	Southwest Airlines	2007	08	52	272
3	Middle Adult	Southwest Airlines	2007	06	93	365
4	Middle Adult	Southwest Airlines	2007	04	158	524
5	Middle Adult	Southwest Airlines	2008	11	420	420
6	Middle Adult	Southwest Airlines	2008	08	78	499
7	Middle Adult	Southwest Airlines	2008	07	83	582
8	Middle Adult	Southwest Airlines	2008	04	193	775
9	Middle Adult	Southwest Airlines	2009	04	294	294

g) Execution plan of new query

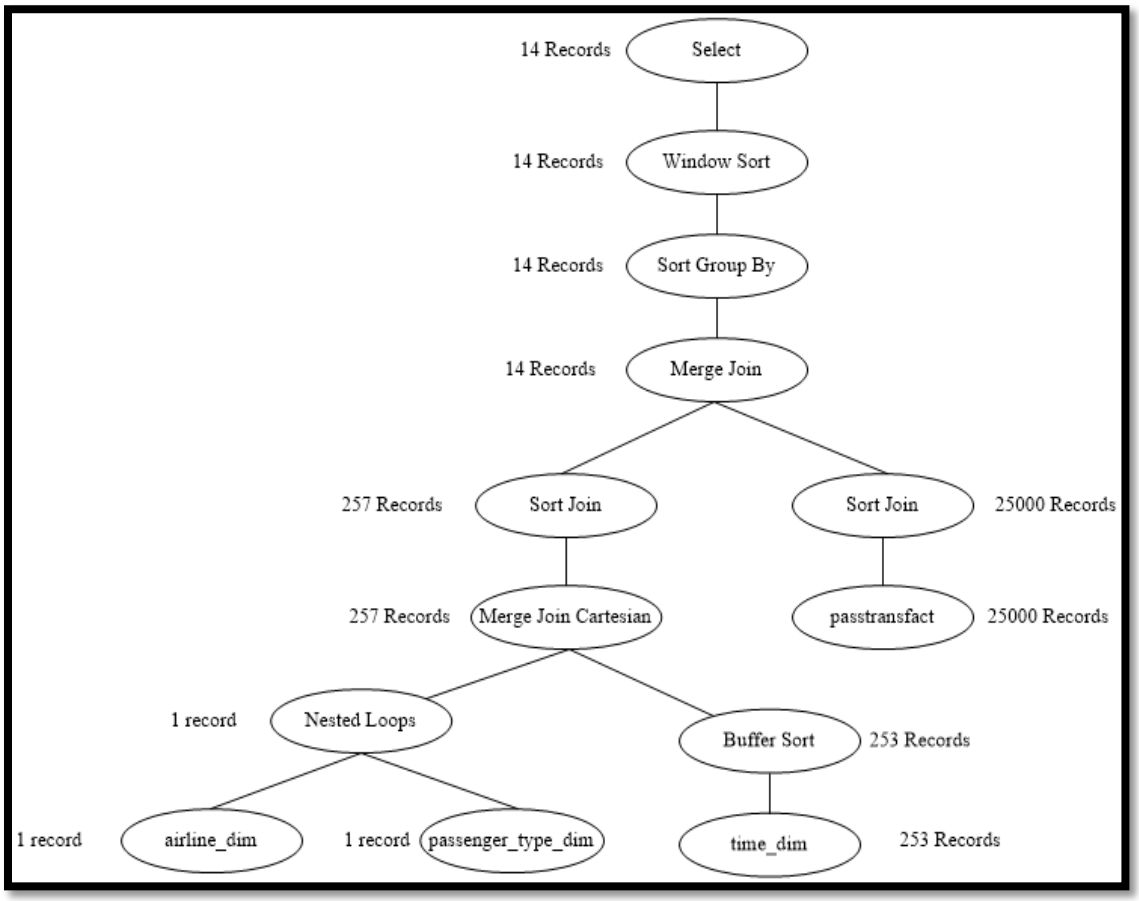
```
Explain Plan for
SELECT /*+ USE_NL(p a) USE_MERGE(f) */ p.PASSTYPEDESC, a.NAME, t.YEARNO,
t.MONTHNAME,
TO_CHAR (SUM(f.TOTALPROFIT), '9,999,999,999') AS PROFIT,
TO_CHAR (SUM(SUM(f.TOTALPROFIT)) OVER
(PARTITION BY t.YEARNO ORDER BY p.PASSTYPEDESC, a.NAME, t.YEARNO,
t.MONTHNAME desc
 ROWS UNBOUNDED PRECEDING),
'9,999,999,999') AS CUM_PROFIT
FROM PASSENGER_TYPE_DIM p, AIRLINE_DIM a, TIME_DIM t, PASSTRANSFACT
f
WHERE p.passtypeid = f.passtypeid
AND t.TIMEID = f.TIMEID
AND a.AIRLINEID = f.AIRLINEID
AND a.NAME = 'Southwest Airlines'
AND p.PASSTYPEDESC = 'Middle Adult'
GROUP BY p.PASSTYPEDESC, a.NAME, t.YEARNO, t.MONTHNAME;

Select * From Table(dbms_xplan.display);
```

Script Output | Query Result

SQL | All Rows Fetched: 29 in 0.019 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 3729123873

-------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                     |    14 |  1274 |       |   305   (2)| 00:00:04 |
|   1 |  WINDOW NOSORT           |                     |    14 |  1274 |       |   305   (2)| 00:00:04 |
|   2 |   SORT GROUP BY          |                     |    14 |  1274 |       |   305   (2)| 00:00:04 |
|   3 |    MERGE JOIN            |                     |    14 |  1274 |       |   304   (2)| 00:00:04 |
|   4 |     SORT JOIN            |                     |   257 | 16191 |       |    21   (5)| 00:00:01 |
|   5 |      MERGE JOIN CARTESIAN|                     |   257 | 16191 |       |    20   (0)| 00:00:01 |
|   6 |       NESTED LOOPS       |                     |     1 |    39 |       |    17   (0)| 00:00:01 |
|* 7  |        TABLE ACCESS FULL | PASSENGER_TYPE_DIM  |     1 |    15 |       |     3   (0)| 00:00:01 |
|* 8  |        TABLE ACCESS FULL | AIRLINE_DIM         |     1 |    24 |       |    14   (0)| 00:00:01 |
|   9 |       BUFFER SORT        |                     |   253 |  6072 |       |     6   (0)| 00:00:01 |
|  10 |        TABLE ACCESS FULL | TIME_DIM            |   253 |  6072 |       |     3   (0)| 00:00:01 |
|* 11 |     SORT JOIN            |                     | 25000 |   683K|  1976K|   282   (1)| 00:00:04 |
|  12 |      TABLE ACCESS FULL   | PASSTRANSFACT       | 25000 |   683K|       |    85   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

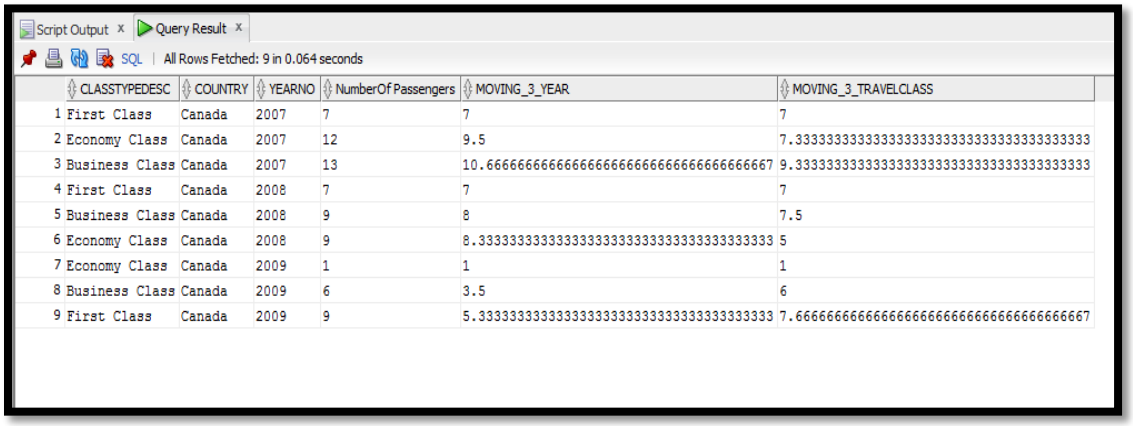
We have provided multiple hints for using nested_loops and use_merge for the updated query execution plan, because the number of records is only 1 for airline_dim and passenger_type_dim and hence nested_loops is a better choice, but the use_merge has introduced extra sorting steps, which are not required, therefore the original query execution plan is better.

3.12. Report 12

a) SQL Query

```
SELECT tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO,
TO_CHAR(SUM(f.TOTALNUMBEROFPASSENGERS)) AS "NumberOf Passengers",
TO_CHAR(AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(PARTITION BY t.YEARNO ORDER BY SUM(f.TOTALNUMBEROFPASSENGERS)
ROWS 2 PRECEDING)) AS MOVING_3_YEAR,
TO_CHAR (AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(partition by tr.classtypedesc ORDER BY sum(f.TOTALNUMBEROFPASSENGERS)
ROWS 2 PRECEDING)) AS MOVING_3_TravelClass
FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, SOURCE_DIM s, TIME_DIM t
WHERE t.TIMEID = f.TIMEID
AND tr.travelid = f.travelid
and s.source_id = f.sourceairportid
AND s.country = 'Canada'
GROUP BY tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO;
```

b) Screenshot of the query result



Script Output | Query Result | SQL | All Rows Fetched: 9 in 0.064 seconds

	CLASSTYPEDESC	COUNTRY	YEARNO	NumberOf Passengers	MOVING_3_YEAR	MOVING_3_TRAVELCLASS
1	First Class	Canada	2007	7	7	7
2	Economy Class	Canada	2007	12	9.5	7.333333333333333333333333333333333333
3	Business Class	Canada	2007	13	10.666666666666666666666666666666666667	9.333333333333333333333333333333333333
4	First Class	Canada	2008	7	7	7
5	Business Class	Canada	2008	9	8	7.5
6	Economy Class	Canada	2008	9	8.333333333333333333333333333333333333	5
7	Economy Class	Canada	2009	1	1	1
8	Business Class	Canada	2009	6	3.5	6
9	First Class	Canada	2009	9	5.333333333333333333333333333333333333	7.666666666666666666666666666666666667

c) Execution plan of original query

```
Explain plan for
SELECT tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO,
```

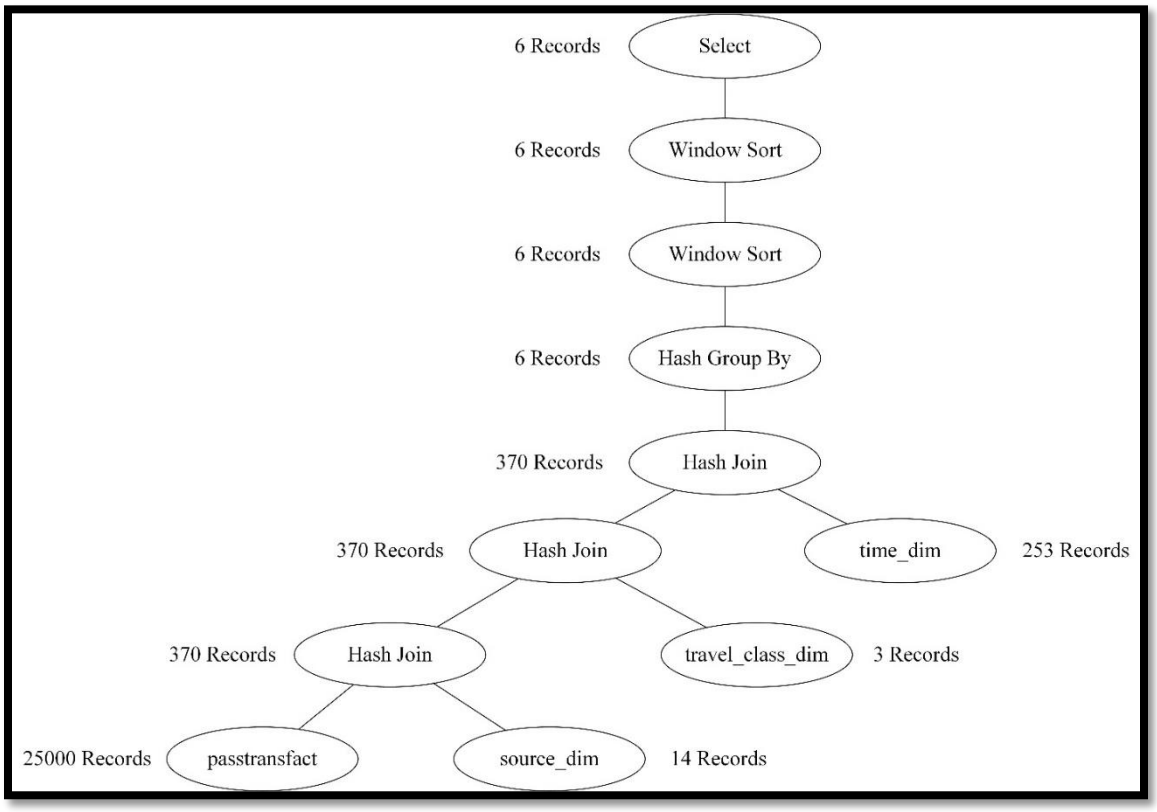
```
TO_CHAR(SUM(f.TOTALNUMBEROFPASSENGERS)) AS "NumberOf Passengers",
TO_CHAR(AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(PARTITION BY t.YEARNO ORDER BY SUM(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_YEAR,
TO_CHAR (AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(partition by tr.classtypedesc ORDER BY sum(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_TravelClass
FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, SOURCE_DIM s, TIME_DIM t
WHERE t.TIMEID = f.TIMEID
AND tr.travelid = f.travelid
and s.source_id = f.sourceairportid
AND s.country = 'Canada'
GROUP BY tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO;
```

Select * from table(dbms_xplan.display);

Script Output | Query Result | SQL | All Rows Fetched: 25 in 0.036 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 3833949954

-----------------------------------------------------------------------------------------------
| Id  | Operation              | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                  |     6 |   468 |   102   (5)| 00:00:02 |
|   1 |  WINDOW SORT           |                  |     6 |   468 |   102   (5)| 00:00:02 |
|   2 |   WINDOW SORT          |                  |     6 |   468 |   102   (5)| 00:00:02 |
|   3 |    HASH GROUP BY       |                  |     6 |   468 |   102   (5)| 00:00:02 |
|*  4 |     HASH JOIN          |                  |   370 | 28860 |    99   (3)| 00:00:02 |
|   5 |      TABLE ACCESS FULL | TIME_DIM         |   253 |  5313 |     3   (0)| 00:00:01 |
|*  6 |      HASH JOIN         |                  |   370 | 21090 |    96   (3)| 00:00:02 |
|   7 |       TABLE ACCESS FULL| TRAVEL_CLASS_DIM |     3 |    51 |     3   (0)| 00:00:01 |
|*  8 |       HASH JOIN        |                  |   370 | 14800 |    92   (2)| 00:00:02 |
|*  9 |        TABLE ACCESS FULL| SOURCE_DIM      |    14 |   196 |     7   (0)| 00:00:01 |
|  10 |        TABLE ACCESS FULL| PASSTRANSFACT   | 25000 |  634K|    84   (0)| 00:00:02 |
-----------------------------------------------------------------------------------------------

```

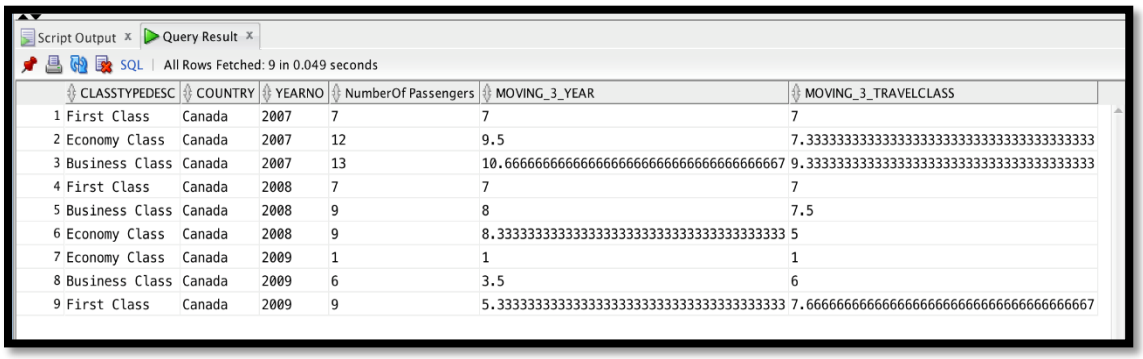
d) Query tree of the original query



e) New SQL

```
SELECT /*+ ORDERED */ tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO,
TO_CHAR(SUM(f.TOTALNUMBEROFPASSENGERS)) AS "NumberOf Passengers",
TO_CHAR(AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(PARTITION BY t.YEARNO ORDER BY SUM(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_YEAR,
TO_CHAR (AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(partition by tr.classtypedesc ORDER BY sum(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_TravelClass
FROM TRAVEL_CLASS_DIM tr, SOURCE_DIM s, TIME_DIM t, PASSTRANSFACT f
WHERE t.TIMEID = f.TIMEID
AND tr.travelid = f.travelid
and s.source_id = f.sourceairportid
AND s.country = 'Canada'
GROUP BY tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO;
```

f) Screenshot of the new query result



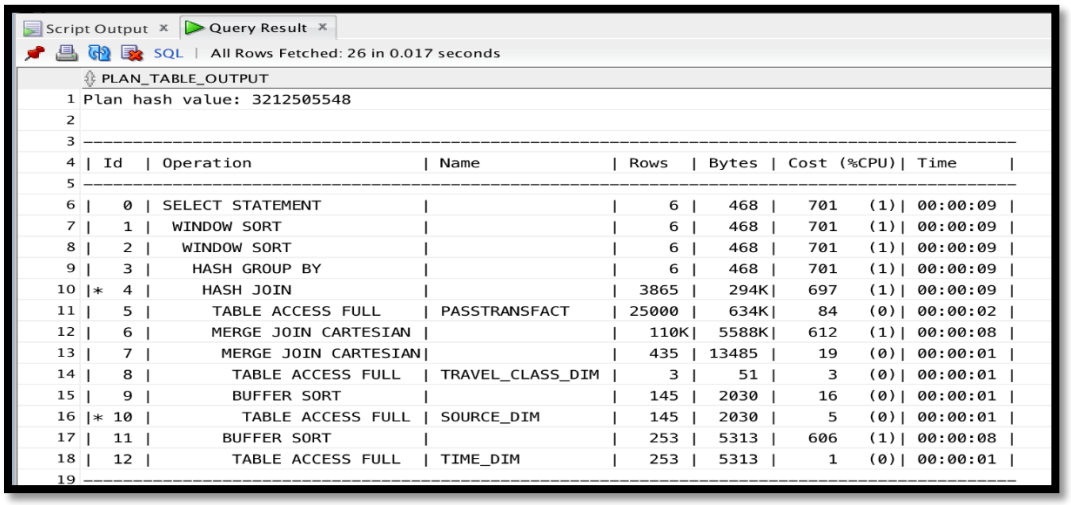
Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.049 seconds

	CLASSTYPEDESC	COUNTRY	YEARNO	NumberOf Passengers	MOVING_3_YEAR	MOVING_3_TRAVELCLASS
1	First Class	Canada	2007	7	7	7
2	Economy Class	Canada	2007	12	9.5	7.333333333333333333333333333333333333333
3	Business Class	Canada	2007	13	10.666666666666666666666666666666666666667	9.333333333333333333333333333333333333333
4	First Class	Canada	2008	7	7	7
5	Business Class	Canada	2008	9	8	7.5
6	Economy Class	Canada	2008	9	8.333333333333333333333333333333333333333	5
7	Economy Class	Canada	2009	1	1	1
8	Business Class	Canada	2009	6	3.5	6
9	First Class	Canada	2009	9	5.333333333333333333333333333333333333333	7.666666666666666666666666666666666666667

g) Execution plan of new query

```
Explain Plan for
SELECT /*+ ORDERED */ tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO,
TO_CHAR(SUM(f.TOTALNUMBEROFPASSENGERS)) AS "NumberOf Passengers",
TO_CHAR(AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(PARTITION BY t.YEARNO ORDER BY SUM(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_YEAR,
TO_CHAR (AVG(SUM(f.TOTALNUMBEROFPASSENGERS)) OVER
(partition by tr.classtypedesc ORDER BY sum(f.TOTALNUMBEROFPASSENGERS)
 ROWS 2 PRECEDING)) AS MOVING_3_TravelClass
FROM TRAVEL_CLASS_DIM tr, SOURCE_DIM s, TIME_DIM t, PASSTRANSFACT f
WHERE t.TIMEID = f.TIMEID
AND tr.travelid = f.travelid
and s.source_id = f.sourceairportid
AND s.country = 'Canada'
GROUP BY tr.CLASSTYPEDESC, s.COUNTRY, t.YEARNO;

Select * From Table(dbms_xplan.display);
```

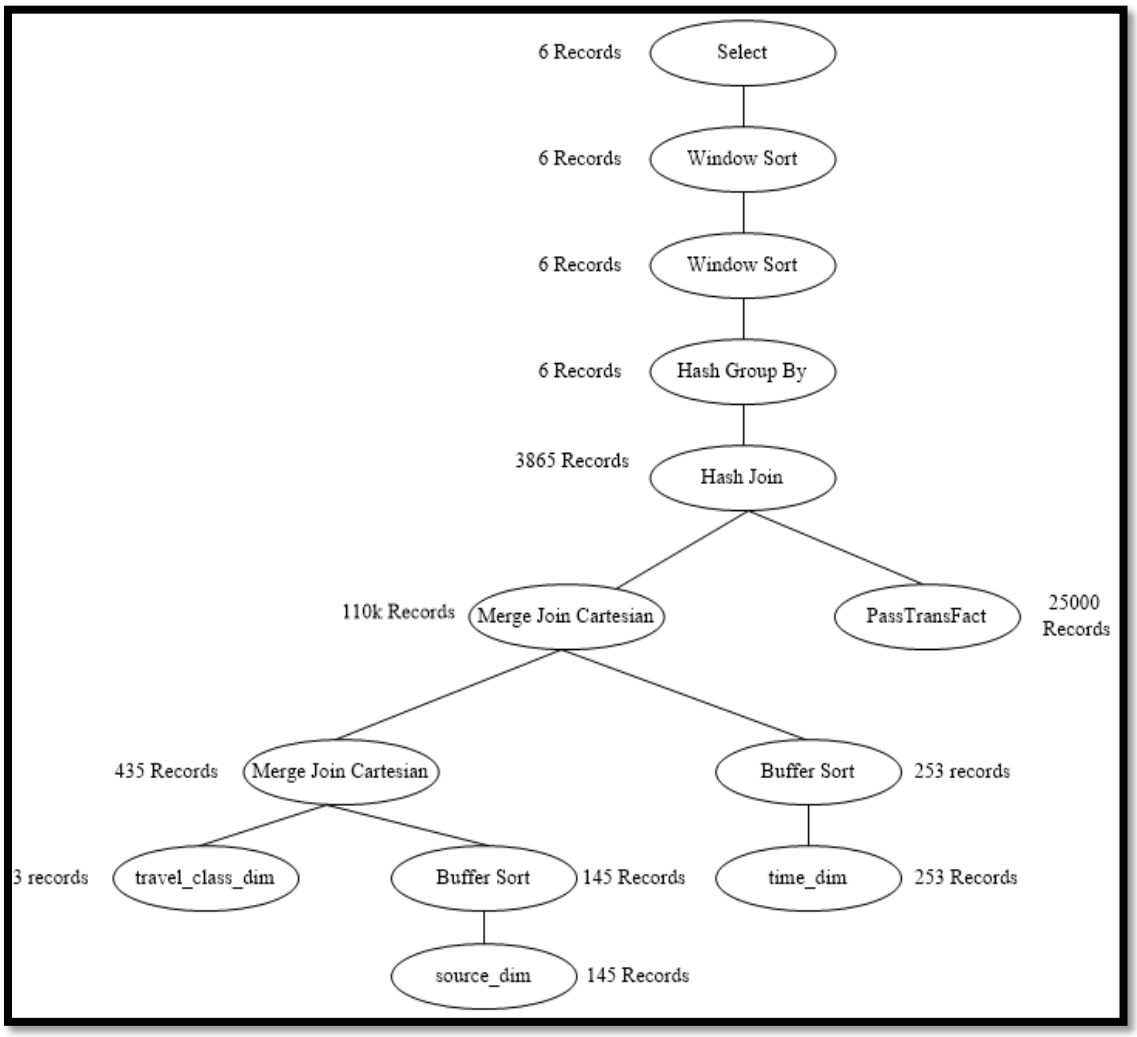


Script Output | Query Result

SQL | All Rows Fetched: 26 in 0.017 seconds

	PLAN_TABLE_OUTPUT
1	Plan hash value: 3212505548
2	
3	--
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	--
6	| 0 | SELECT STATEMENT | | 6 | 468 | 701 (1)| 00:00:09 |
7	| 1 | WINDOW SORT | | 6 | 468 | 701 (1)| 00:00:09 |
8	| 2 | WINDOW SORT | | 6 | 468 | 701 (1)| 00:00:09 |
9	| 3 | HASH GROUP BY | | 6 | 468 | 701 (1)| 00:00:09 |
10	|* 4 | HASH JOIN | | 3865 | 294K| 697 (1)| 00:00:09 |
11	| 5 | TABLE ACCESS FULL | PASSTRANSFACT | 25000 | 634K| 84 (0)| 00:00:02 |
12	| 6 | MERGE JOIN CARTESIAN | | 110K| 5588K| 612 (1)| 00:00:08 |
13	| 7 | MERGE JOIN CARTESIAN| | 435 | 13485 | 19 (0)| 00:00:01 |
14	| 8 | TABLE ACCESS FULL | TRAVEL_CLASS_DIM | 3 | 51 | 3 (0)| 00:00:01 |
15	| 9 | BUFFER SORT | | 145 | 2030 | 16 (0)| 00:00:01 |
16	|* 10 | TABLE ACCESS FULL | SOURCE_DIM | 145 | 2030 | 5 (0)| 00:00:01 |
17	| 11 | BUFFER SORT | | 253 | 5313 | 606 (1)| 00:00:08 |
18	| 12 | TABLE ACCESS FULL | TIME_DIM | 253 | 5313 | 1 (0)| 00:00:01 |
19	--

h) Query tree of the new query



i) Explanation on why one query is better than the other

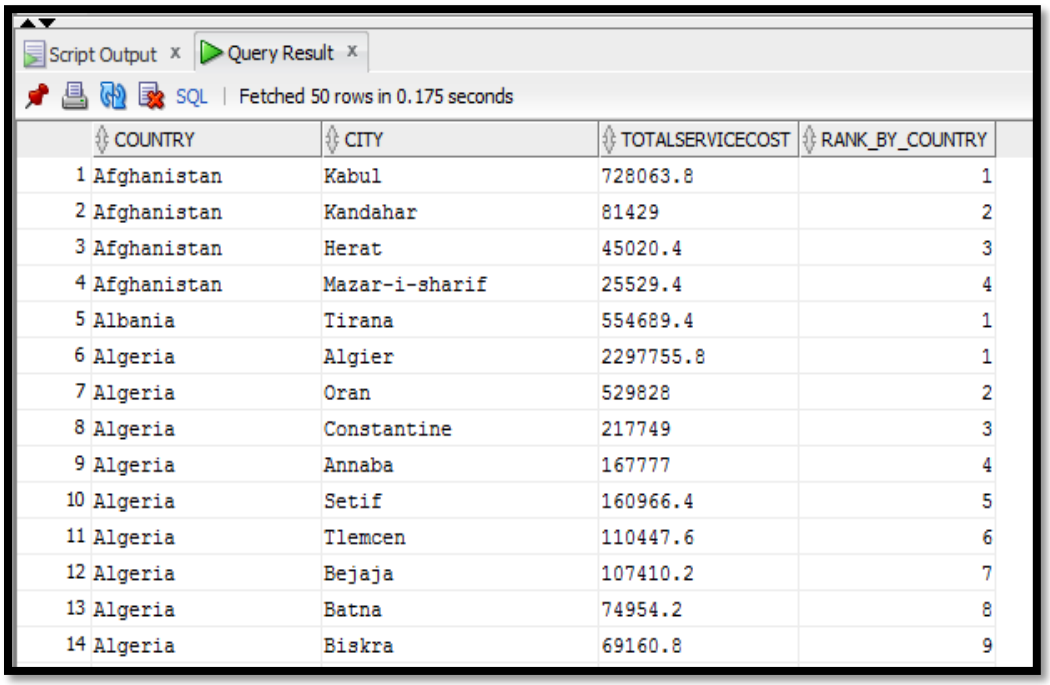
We have added a hint of ordered to alter the execution plan, and have mentioned the fact table as the least important table in the order of execution because the number of records is 25000 which is much more than the records in dimension tables. But the ordered hint resulted in 110K records after the merge join Cartesian of the dimension tables itself, which is much more than the records of the fact table, 25000. So the original query execution plan is better.

3.13. Report 13

a) SQL Query

```
SELECT s.country,s.city,
TO_CHAR(SUM(TotalServiceCost)) AS TotalServiceCost,
RANK() OVER (PARTITION BY s.country
ORDER BY SUM(TotalServiceCost) DESC) AS RANK_BY_COUNTRY
FROM source_dim s, routesfact r
WHERE s.source_id=r.sourceairportid
GROUP BY s.country,s.city;
```

b) Screenshot of the query result



Script Output | Query Result | SQL | Fetched 50 rows in 0.175 seconds

	COUNTRY	CITY	TOTALSERVICECOST	RANK_BY_COUNTRY
1	Afghanistan	Kabul	728063.8	1
2	Afghanistan	Kandahar	81429	2
3	Afghanistan	Herat	45020.4	3
4	Afghanistan	Mazar-i-sharif	25529.4	4
5	Albania	Tirana	554689.4	1
6	Algeria	Algier	2297755.8	1
7	Algeria	Oran	529828	2
8	Algeria	Constantine	217749	3
9	Algeria	Annaba	167777	4
10	Algeria	Setif	160966.4	5
11	Algeria	Tlemcen	110447.6	6
12	Algeria	Bejaja	107410.2	7
13	Algeria	Batna	74954.2	8
14	Algeria	Biskra	69160.8	9

c) Execution plan of original query

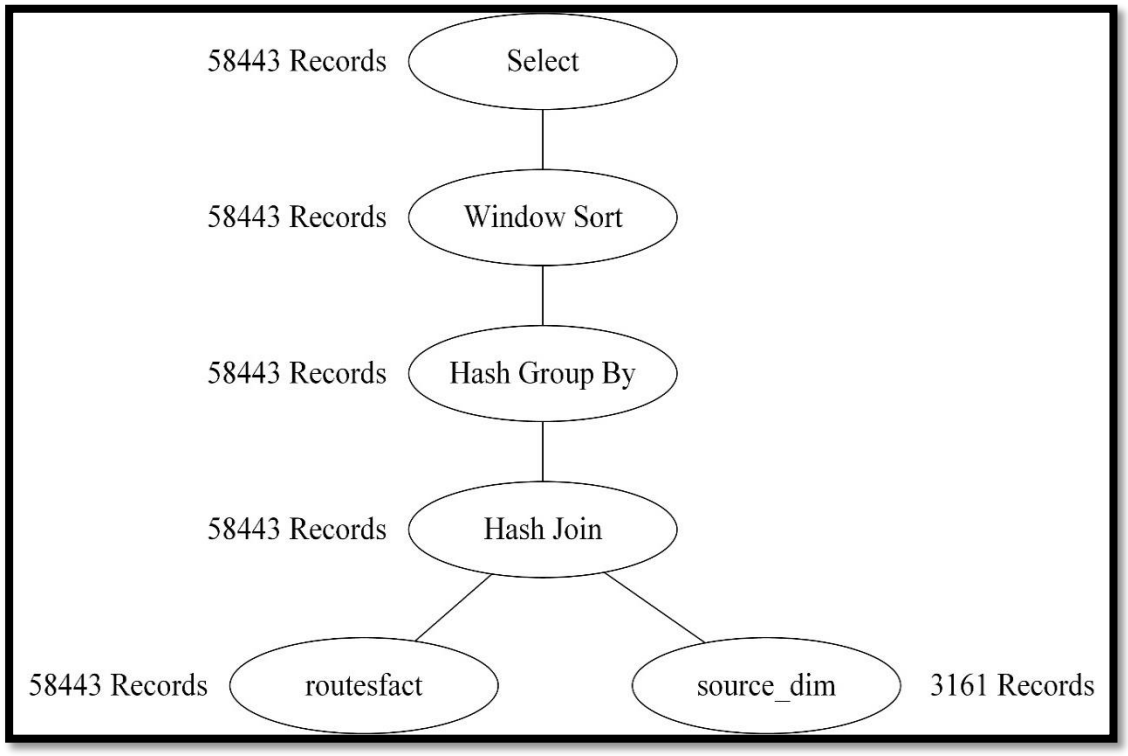
```
Explain plan for
SELECT s.country,s.city,
TO_CHAR(SUM(TotalServiceCost)) AS TotalServiceCost,
RANK() OVER (PARTITION BY s.country
ORDER BY SUM(TotalServiceCost) DESC) AS RANK_BY_COUNTRY
FROM source_dim s, routesfact r
WHERE s.source_id=r.sourceairportid
GROUP BY s.country,s.city;
```

Select * from table(dbms_xplan.display);

Script Output | Query Result | SQL | All Rows Fetched: 17 in 0.033 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 2913362706

-----------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             | 58443 |  1940K|       |  1152   (1)| 00:00:14 |
|   1 |  WINDOW SORT         |             | 58443 |  1940K|  2536K|  1152   (1)| 00:00:14 |
|   2 |   HASH GROUP BY      |             | 58443 |  1940K|  2536K|  1152   (1)| 00:00:14 |
|*  3 |    HASH JOIN         |             | 58443 |  1940K|       |    82   (3)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| SOURCE_DIM  |  3161 | 75864 |       |     7   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| ROUTESFACT  | 58443 |   570K|       |    74   (2)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

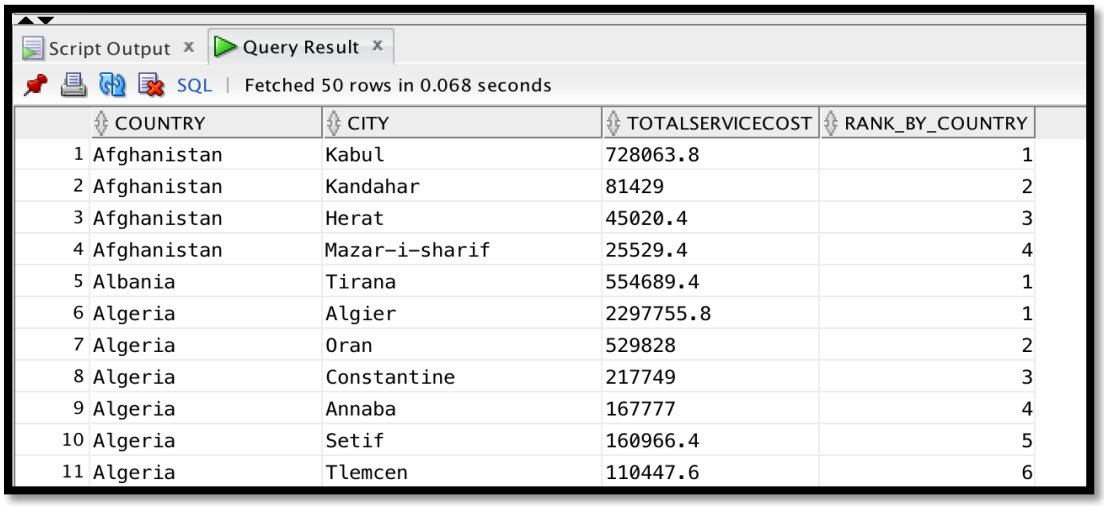
d) Query tree of the original query



e) New SQL

```
SELECT /*+ USE_MERGE(s r) */s.country,s.city,
TO_CHAR(SUM(TotalServiceCost)) AS TotalServiceCost,
RANK() OVER (PARTITION BY s.country
ORDER BY SUM(TotalServiceCost) DESC) AS RANK_BY_COUNTRY
FROM source_dim s, routesfact r
WHERE s.source_id=r.sourceairportid
GROUP BY s.country,s.city;
```

f) Screenshot of the new query result



Script Output | Query Result

SQL | Fetched 50 rows in 0.068 seconds

	COUNTRY	CITY	TOTALSERVICECOST	RANK_BY_COUNTRY
1	Afghanistan	Kabul	728063.8	1
2	Afghanistan	Kandahar	81429	2
3	Afghanistan	Herat	45020.4	3
4	Afghanistan	Mazar-i-sharif	25529.4	4
5	Albania	Tirana	554689.4	1
6	Algeria	Algier	2297755.8	1
7	Algeria	Oran	529828	2
8	Algeria	Constantine	217749	3
9	Algeria	Annaba	167777	4
10	Algeria	Setif	160966.4	5
11	Algeria	Tlemcen	110447.6	6

g) Execution plan of new query

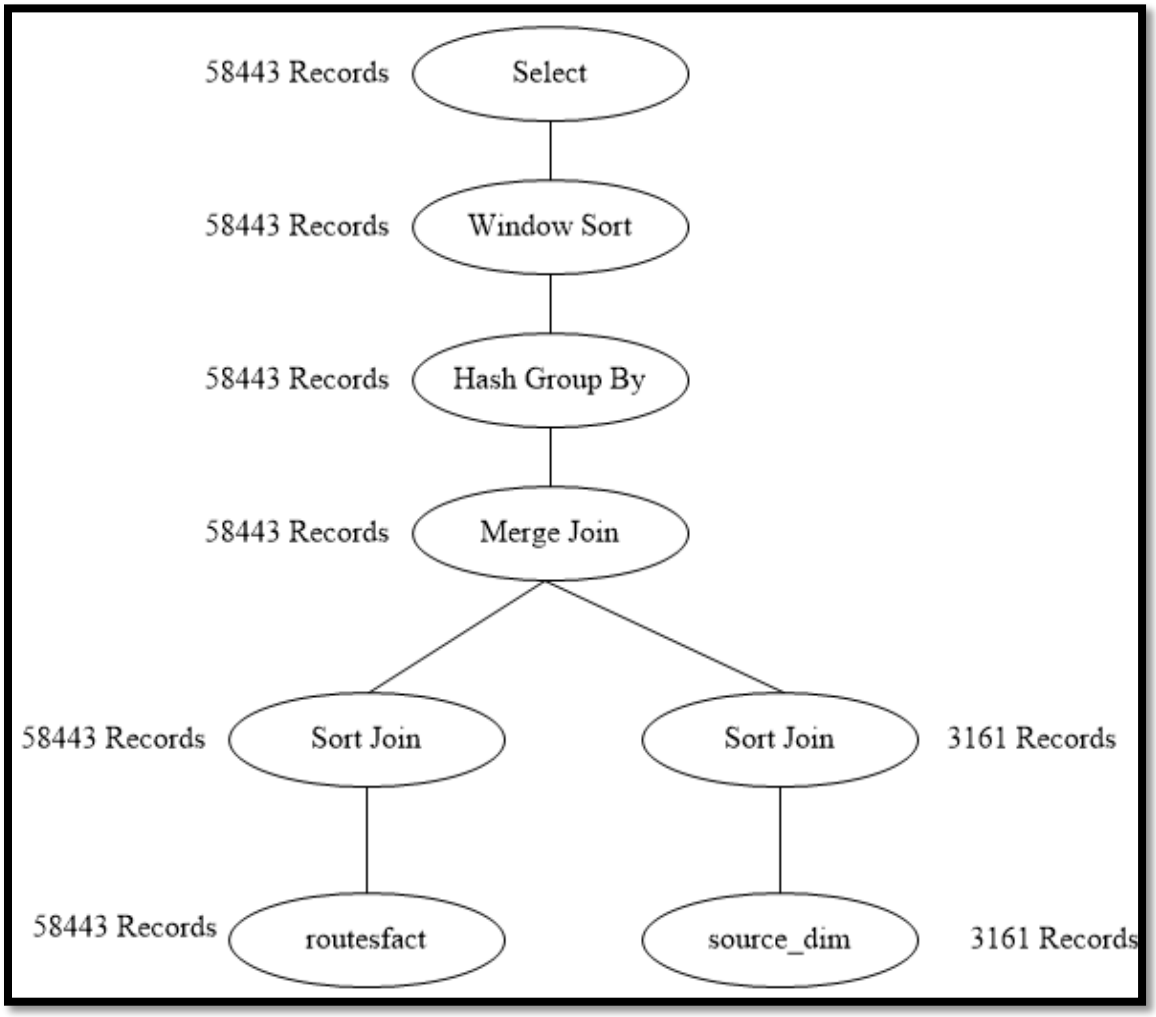
```
Explain Plan for
SELECT /*+ USE_MERGE(s r) */s.country,s.city,
TO_CHAR(SUM(TotalServiceCost)) AS TotalServiceCost,
RANK() OVER (PARTITION BY s.country
ORDER BY SUM(TotalServiceCost) DESC) AS RANK_BY_COUNTRY
FROM source_dim s, routesfact r
WHERE s.source_id=r.sourceairportid
GROUP BY s.country,s.city;

Select * From Table(dbms_xplan.display);
```

Script Output × Query Result ×

SQL | All Rows Fetched: 20 in 0.017 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 3723736005

-------------------------------------------------------------------------------------------------
| Id  | Operation             | Name        | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             | 58443 |  1940K|       |  1391   (1)| 00:00:17 |
|   1 |  WINDOW SORT          |             | 58443 |  1940K|  2536K|  1391   (1)| 00:00:17 |
|   2 |   HASH GROUP BY       |             | 58443 |  1940K|  2536K|  1391   (1)| 00:00:17 |
|   3 |    MERGE JOIN         |             | 58443 |  1940K|       |   320   (2)| 00:00:04 |
|   4 |     SORT JOIN         |             |  3161 | 75864 |       |     8  (13)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| SOURCE_DIM  |  3161 | 75864 |       |     7   (0)| 00:00:01 |
|*  6 |     SORT JOIN         |             | 58443 |   570K|  2312K|   312   (2)| 00:00:04 |
|   7 |      TABLE ACCESS FULL| ROUTESFACT  | 58443 |   570K|       |    74   (2)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

h) Query tree of the new query



i) Explanation on why one query is better than the other

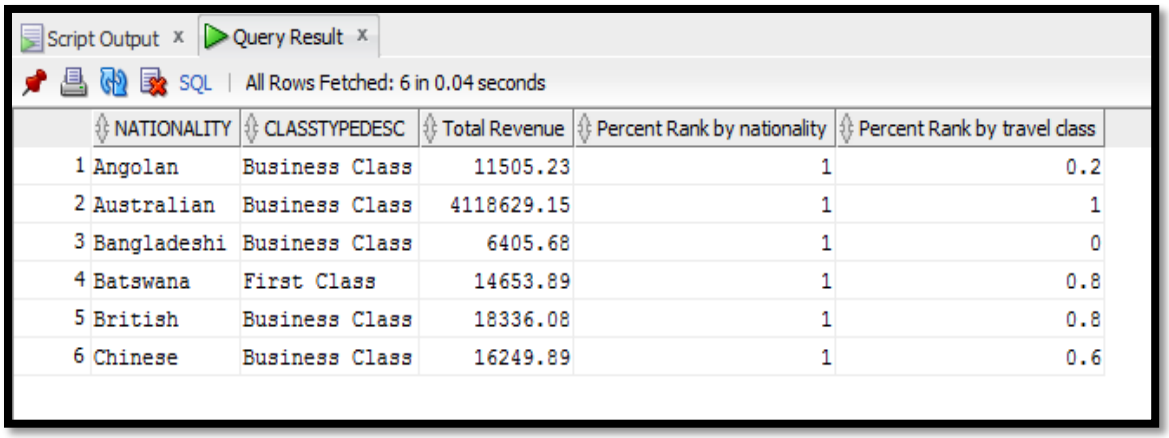
We have provided the hint of use_merge, which has introduced an extra operation of sorting at the very beginning in the execution plan, which is not efficient and uses more CPU and time. The original query execution plan is efficient as the number of records are large for source_dim and RoutesFact which are 3161 and 58443 records respectively. Since the number of records is higher, hash join should be performed.

3.14. Report 14:

a) SQL Query

```
select *
from (
SELECT
n.NATIONALITY, tr.classtypedesc,
sum(f.SUMOFTOTALPAID) as "Total Revenue",
percent_rank() over
(partition by n.nationality order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by nationality",
percent_rank() over
(partition by tr.classtypedesc order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by travel class"
FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, NATIONALITY_DIM n
WHERE tr.TRAVELID = f.TRAVELID
and n.NATIONALITY = f.NATIONALITY
and n.NATIONALITY IN ('Angolan','Australian','British','Bangladeshi','Chinese','Batswana')
GROUP BY n.NATIONALITY, tr.classtypedesc
) where "Percent Rank by nationality" >= 0.9;
```

b) Screenshot of the query result


Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.04 seconds

	NATIONALITY	CLASSTYPEDESC	Total Revenue	Percent Rank by nationality	Percent Rank by travel class
1	Angolan	Business Class	11505.23	1	0.2
2	Australian	Business Class	4118629.15	1	1
3	Bangladeshi	Business Class	6405.68	1	0
4	Batswana	First Class	14653.89	1	0.8
5	British	Business Class	18336.08	1	0.8
6	Chinese	Business Class	16249.89	1	0.6

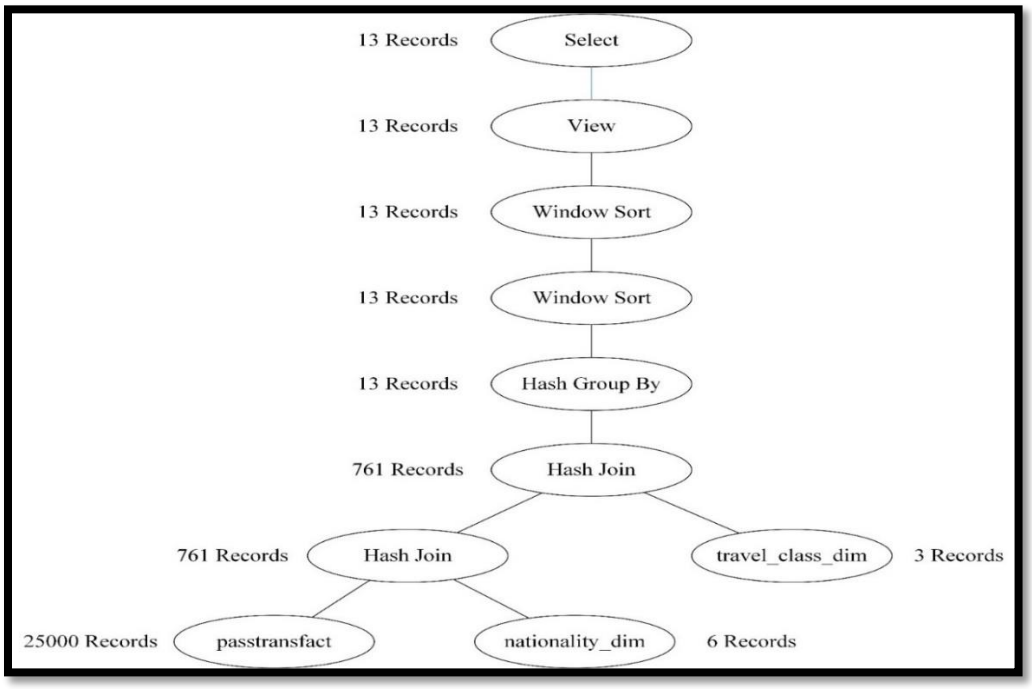
c) Execution plan of original query

```
Explain plan for
select *
from (
SELECT
n.NATIONALITY, tr.classtypedesc,
sum(f.SUMOFTOTALPAID) as "Total Revenue",
percent_rank() over
(partition by n.nationality order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by nationality",
percent_rank() over
(partition by tr.classtypedesc order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by travel class"
FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, NATIONALITY_DIM n
WHERE tr.TRAVELID = f.TRAVELID
and n.NATIONALITY = f.NATIONALITY
and n.NATIONALITY IN ('Angolan','Australian','British','Bangladeshi','Chinese','Batswana')
GROUP BY n.NATIONALITY, tr.classtypedesc
) where "Percent Rank by nationality" >= 0.9;

Select * from table(dbms_xplan.display);
```

PLAN_TABLE_OUTPUT

```
Plan hash value: 2775070813

-----------------------------------------------------------------------------------------------
| Id  | Operation              | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                  |    13 |   845 |    95   (6)| 00:00:02 |
|*  1 |  VIEW                  |                  |    13 |   845 |    95   (6)| 00:00:02 |
|   2 |   WINDOW SORT          |                  |    13 |   598 |    95   (6)| 00:00:02 |
|   3 |    WINDOW SORT         |                  |    13 |   598 |    95   (6)| 00:00:02 |
|   4 |     HASH GROUP BY      |                  |    13 |   598 |    95   (6)| 00:00:02 |
|*  5 |      HASH JOIN         |                  |   761 | 35006 |    92   (3)| 00:00:02 |
|   6 |       TABLE ACCESS FULL| TRAVEL_CLASS_DIM |     3 |    51 |     3   (0)| 00:00:01 |
|*  7 |       HASH JOIN        |                  |   761 | 22069 |    88   (2)| 00:00:02 |
|*  8 |        TABLE ACCESS FULL| NATIONALITY_DIM |     6 |    60 |     3   (0)| 00:00:01 |
|*  9 |        TABLE ACCESS FULL| PASSTRANSFACT   | 25000 |  463K|    85   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------

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

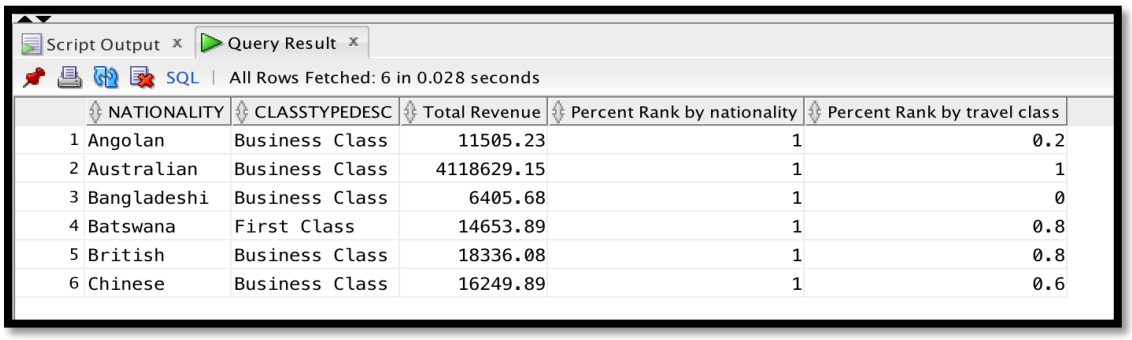
d) Query tree of the original query



e) New SQL

```
select *
from (
SELECT /*+ USE_MERGE (n tr) */n.NATIONALITY, tr.classtypedesc,
sum(f.SUMOFTOTALPAID) as "Total Revenue",
percent_rank() over
(partition by n.nationality order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by
nationality",
percent_rank() over
(partition by tr.classtypedesc order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by
travel class"
FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, NATIONALITY_DIM n
WHERE tr.TRAVELID = f.TRAVELID
and n.NATIONALITY = f.NATIONALITY
and n.NATIONALITY IN ('Angolan','Australian','British','Bangladeshi','Chinese','Batswana')
GROUP BY n.NATIONALITY, tr.classtypedesc
) where "Percent Rank by nationality" >= 0.9;
```

f) Screenshot of the new query result



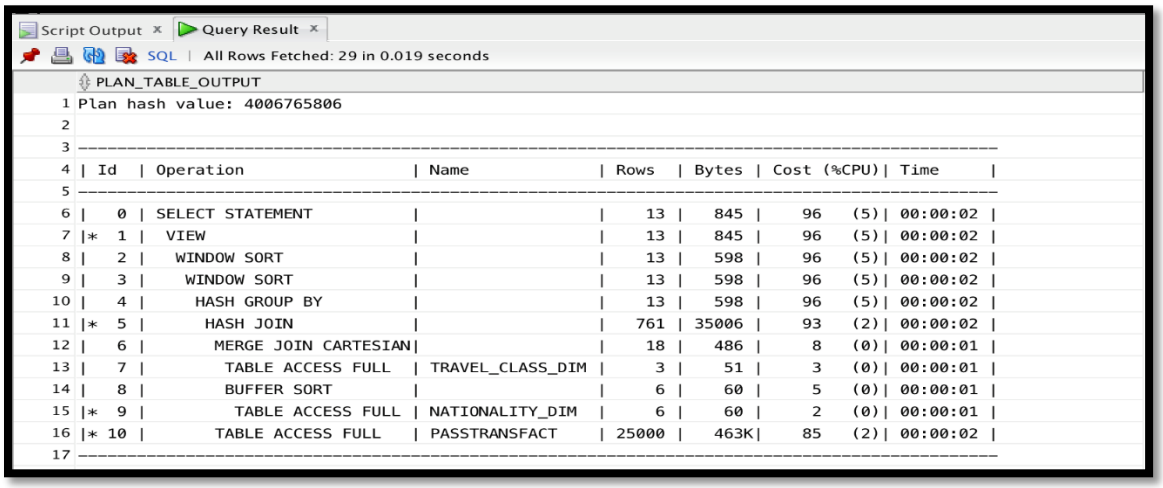
Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.028 seconds

	NATIONALITY	CLASSTYPEDESC	Total Revenue	Percent Rank by nationality	Percent Rank by travel class
1	Angolan	Business Class	11505.23	1	0.2
2	Australian	Business Class	4118629.15	1	1
3	Bangladeshi	Business Class	6405.68	1	0
4	Batswana	First Class	14653.89	1	0.8
5	British	Business Class	18336.08	1	0.8
6	Chinese	Business Class	16249.89	1	0.6

g) Execution plan of new query

```
Explain Plan for
select *
from (
SELECT /*+ USE_MERGE (n tr) */n.NATIONALITY, tr.classtypedesc,
sum(f.SUMOFTOTALPAID) as "Total Revenue",
percent_rank() over
(partition by n.nationality order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by
nationality",
percent_rank() over
(partition by tr.classtypedesc order by sum(f.SUMOFTOTALPAID)) as "Percent Rank by
travel class"
FROM PASSTRANSFACT f, TRAVEL_CLASS_DIM tr, NATIONALITY_DIM n
WHERE tr.TRAVELID = f.TRAVELID
and n.NATIONALITY = f.NATIONALITY
and n.NATIONALITY IN ('Angolan','Australian','British','Bangladeshi','Chinese','Batswana')
GROUP BY n.NATIONALITY, tr.classtypedesc
) where "Percent Rank by nationality" >= 0.9;

Select * From Table(dbms_xplan.display);
```

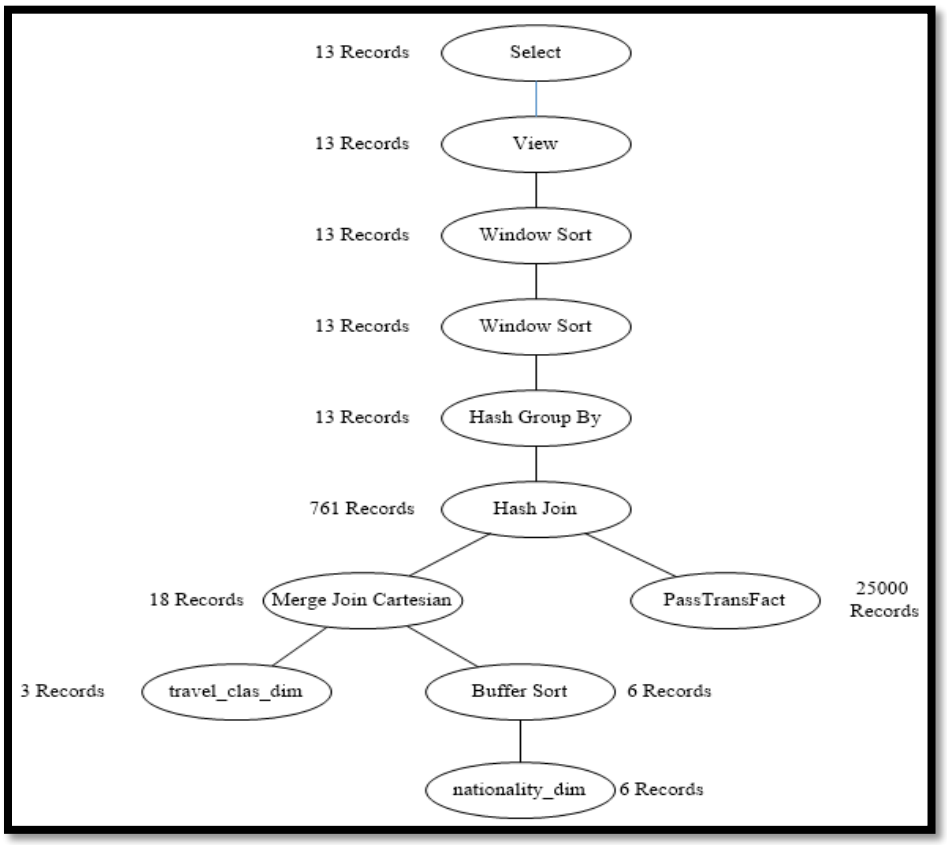


Script Output | Query Result

SQL | All Rows Fetched: 29 in 0.019 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 4006765806

-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |    13 |   845 |    96   (5)| 00:00:02 |
|*  1 |  VIEW                        |                   |    13 |   845 |    96   (5)| 00:00:02 |
|   2 |   WINDOW SORT                |                   |    13 |   598 |    96   (5)| 00:00:02 |
|   3 |    WINDOW SORT               |                   |    13 |   598 |    96   (5)| 00:00:02 |
|   4 |     HASH GROUP BY            |                   |    13 |   598 |    96   (5)| 00:00:02 |
|*  5 |      HASH JOIN               |                   |   761 | 35006 |    93   (2)| 00:00:02 |
|   6 |       MERGE JOIN CARTESIAN   |                   |    18 |   486 |     8   (0)| 00:00:01 |
|   7 |        TABLE ACCESS FULL     | TRAVEL_CLASS_DIM  |     3 |    51 |     3   (0)| 00:00:01 |
|   8 |        BUFFER SORT           |                   |     6 |    60 |     5   (0)| 00:00:01 |
|*  9 |         TABLE ACCESS FULL    | NATIONALITY_DIM   |     6 |    60 |     2   (0)| 00:00:01 |
|* 10 |       TABLE ACCESS FULL      | PASSTRANSFACT     | 25000 |  463K|    85   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------------
```

h) Query tree of the new query

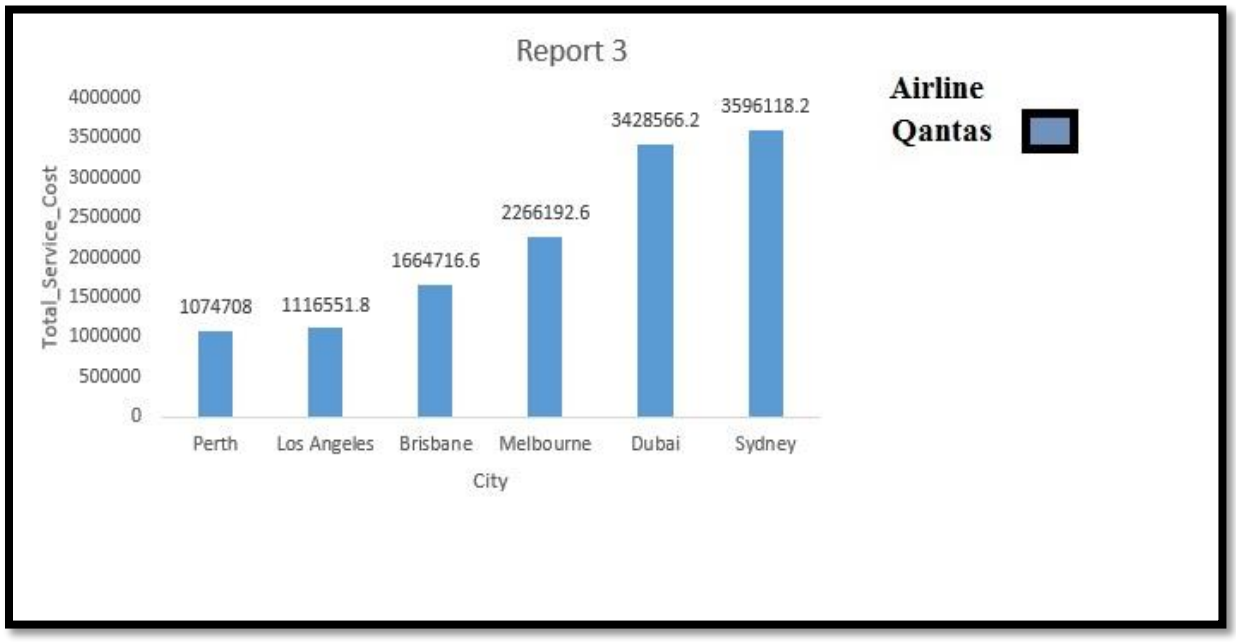


i) Explanation on why one query is better than the other

The new query execution plan is better because we have pushed the baggage of PassTransFact records of 25000 at a later stage, which saves the CPU effort and time of execution.

4. Task C.4

Report 3:



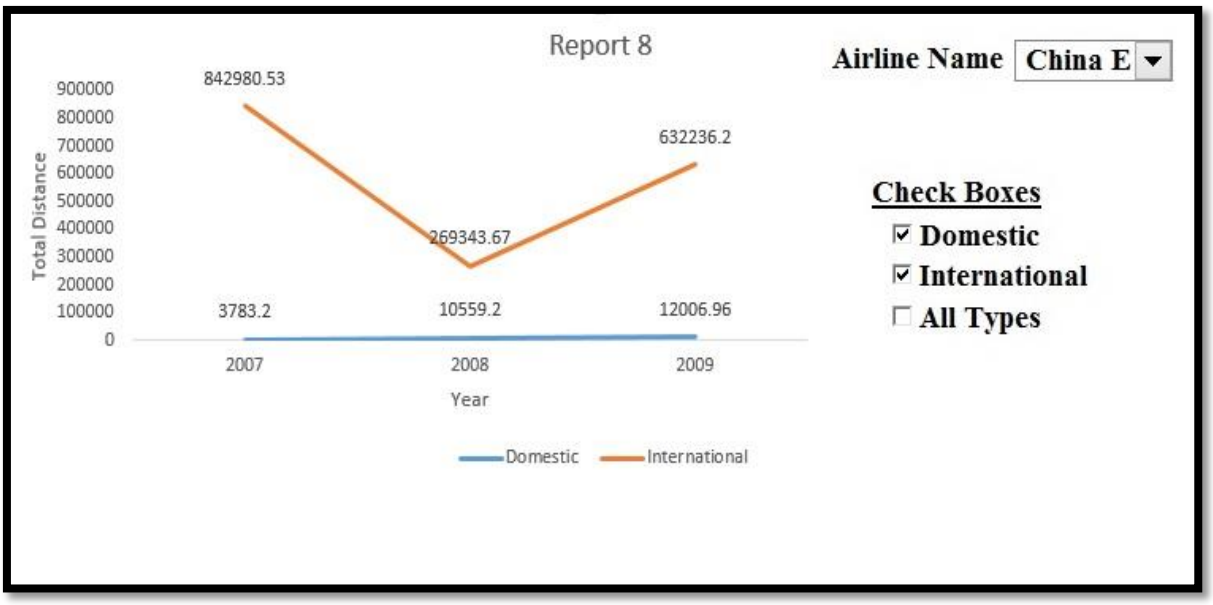
The above graph is plotted for top 5% total service cost against the cities. The airline legend shows the airline that has the total service cost. In our case all the top 5 service cost belong to Qantas Airline hence the color of all bars is blue and the same is visible in legend.

Report 7:

The above graph is plotted for Total Revenue generated VS Flight Type. There are three combo box named Month, Year and Flight Type. In Month combo box months can be selected as well as there is an option of selecting 'All Month'. In Year combo box all the valid years can be selected

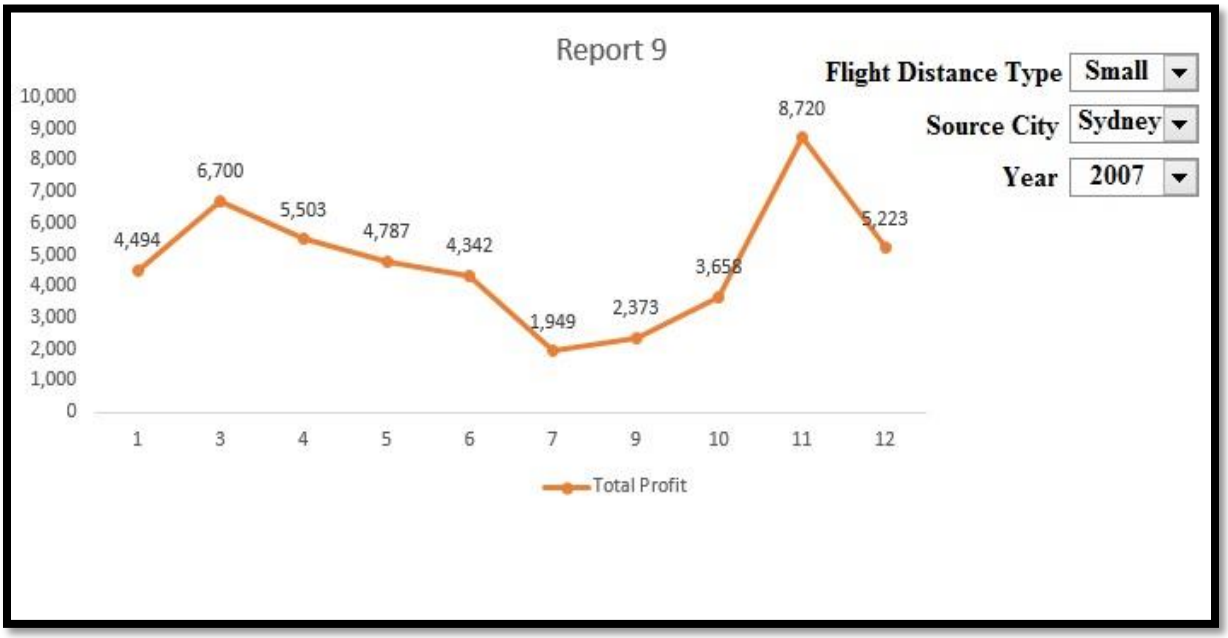
as well as there is an option of selecting 'All Year'. The Flight Type combo box contains three options of Domestic, International and All Flight Type.

Report 8:



The above graph is plotted for Total Distance VS Year. There is Airline Name combo box which contains individual airline names as well as All Airlines option. Check boxes are also provided to select line graph of Domestic, International or All Types.

Report 9:



The above graph is plotted for Monthly Total Profit. There are 3 combo boxes Flight Distance Type, Source City and Year. Flight Distance Type contains 5 options Small, Medium, Large, Very Large and All Distance type. Source City contains all source cities and All Source City option. Year Combo box contains valid years to select from.

5. Task C.5

Business Intelligence Systems are the systems which provide complex internal and competitive information to key personnel, planners and decision makers by collecting, storing and combining data using analytical tools. Therefore the main objective of business information systems is to facilitate managerial work by improving the timeliness and quality of inputs to the decision process (Negash, 2004). In order to perform properly Business Intelligence Systems require a proper Multidimensional Data Warehouse.

Applications of Business Information Systems (Chen, Chiang, & Storey, 2012)

- Reporting
- Dashboards
- Interactive Visualizations
- Data mining
- Forecasting

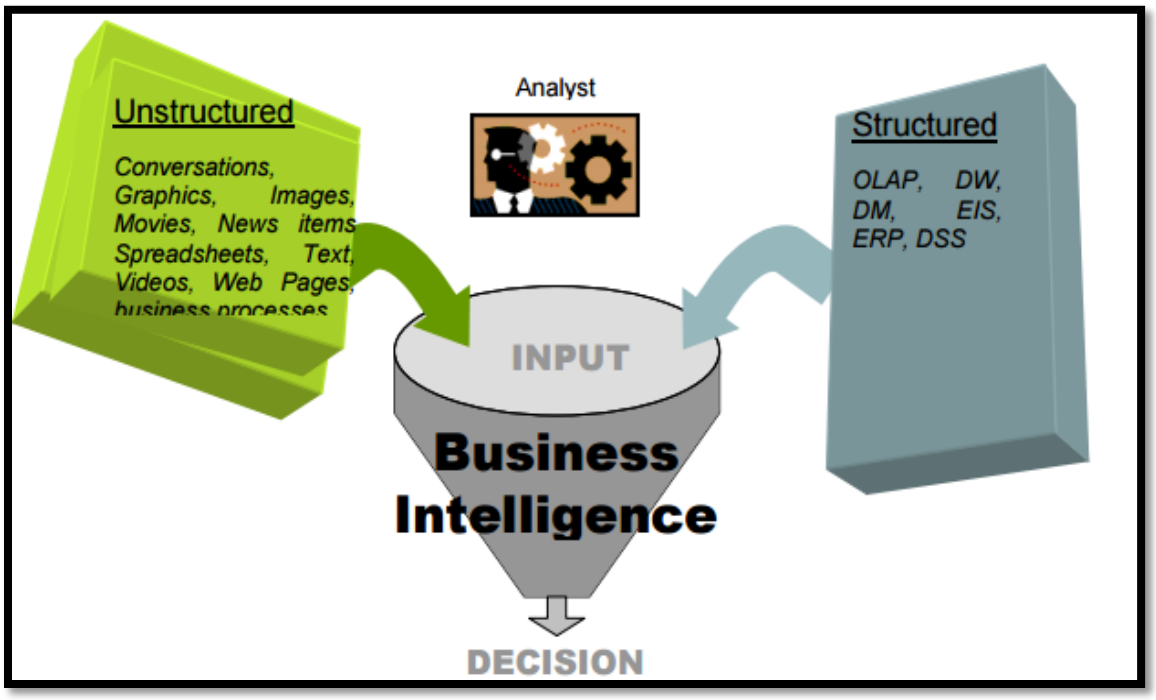


Figure1: Business Intelligence (Negash, 2004)

Data Warehouse can be defined as the database which is responsible for collecting consistent, subject-oriented, integrated, time-variant and non-volatile data which can be analyzed in order to support management decision making. Using a data warehouse is involves three phases:

- First step involves extracting information from multiple transaction processing systems.
- Second step involves transforming, aggregating and cleaning information and then storing it into data warehouse.
- Third Step involves accessing integrated data in an efficient and flexible manner.

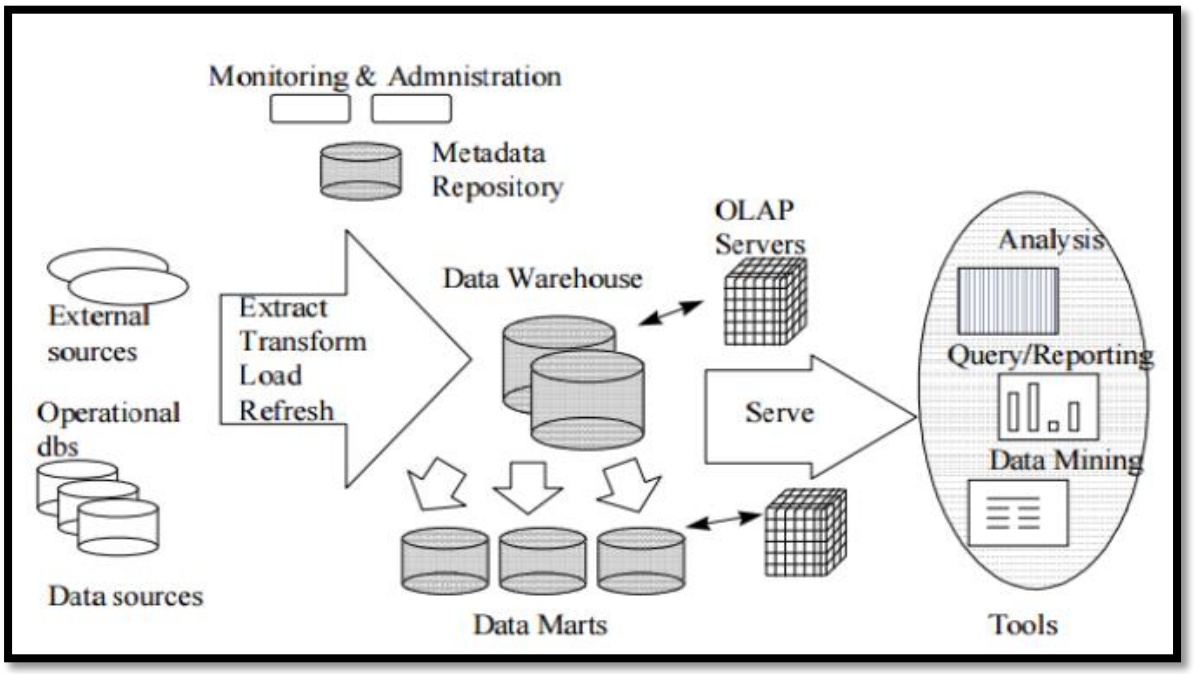


Figure 2: Data Warehouse Architecture (Chaudhuri & Dayal, 1997)

Developing efficient Data Warehouse is of utmost important to organizations because it increases the capability of converting collected data into strategic information which can be made accessible to decision makers. Online Analytical Processing is the capability which provides end users with configurable views of data from different perspectives, angles and on different aggregation levels. Further in order to run fast and flexible OLAP queries data must be arranged in multidimensional form which means storing information in the form of facts and dimension. Facts are numeric, factual data representing a business activity while dimensions are single perspective of data that determines granularity (Paim & de Castro, 2003).

Consider the following example of MacDonald's Restaurant which has the following transaction processing systems.

The database used by restaurant for the following:

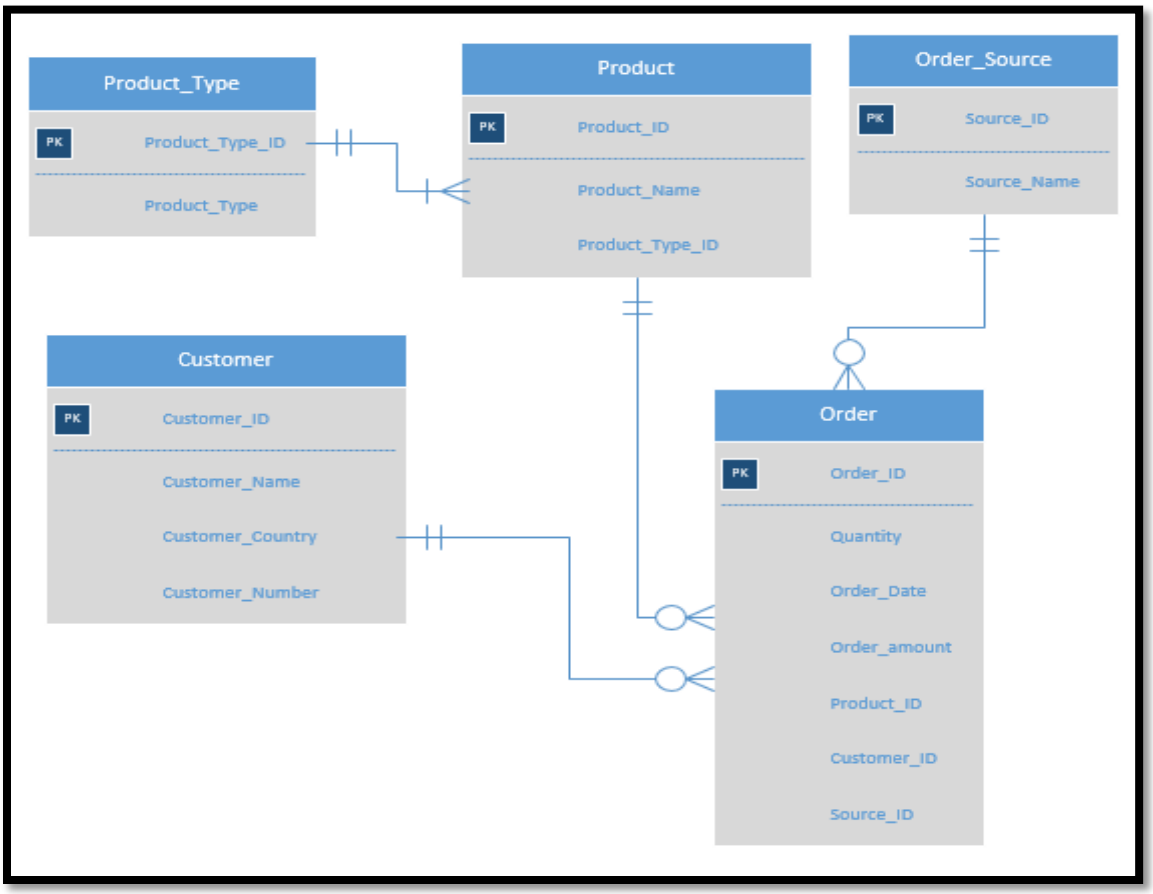


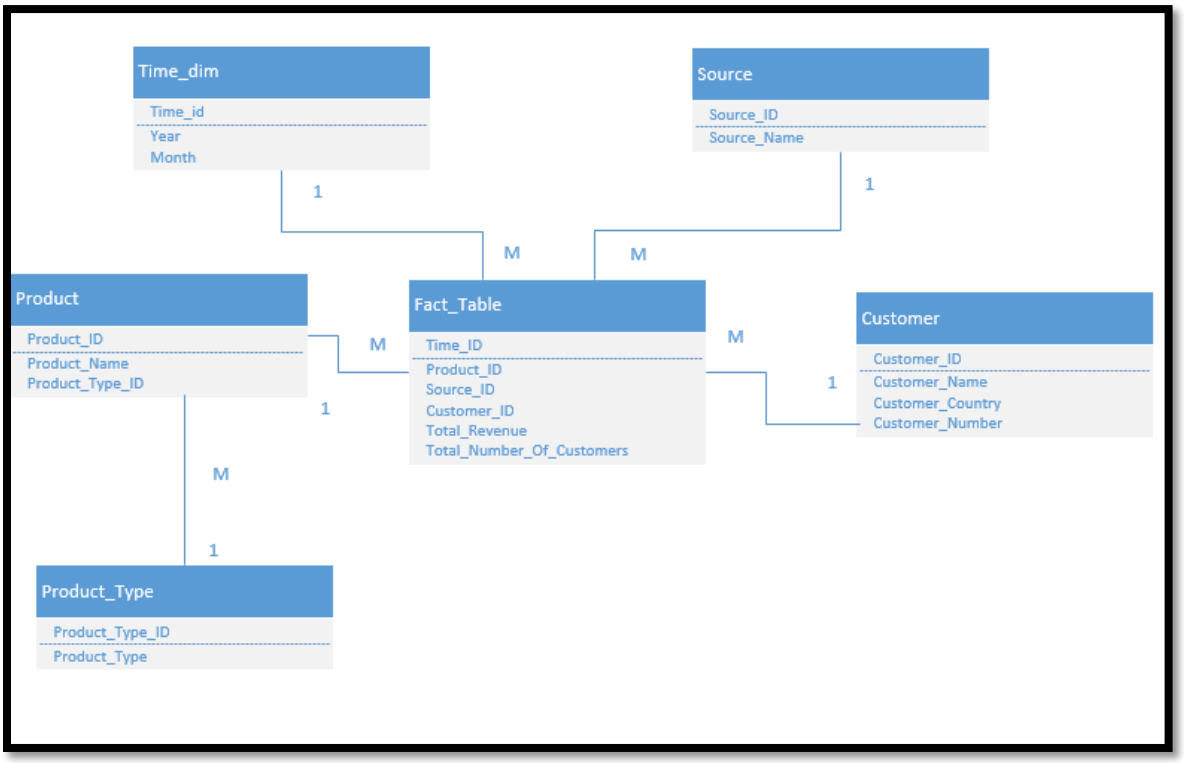
Table	Use
Product	Various products which the restaurant sells
Product type	Classifies the product into various categories
Order source	Contains the data how the order was given by the customers like online, drive through or dine in
Customer	Contains the customer details
Order type	Various order details.

Now consider the management is interested in knowing the total order revenue or total number of customers with respect to year, month, product name, source name, customer etc. The above results are possible using transaction processing systems but there are some issues like it requires complex queries which are difficult to write, requires multiple joins which degrades performance and makes it slow and the transaction process system upgrades frequently.

Due to above issues there was a rise of multidimensional database system. In Multidimensional database systems the data from transaction processing systems is extracted, cleaned and then

uploaded into dimension and fact table. The advantages are simple queries are required to extract data as there are fewer joins and performance is fast. Moreover the multidimensional database is not often upgraded which gives current results. Moreover if the above described restaurant had different transactional process for online, dine in and drive through we could combine the data in dimensional database.

The multidimensional data base for the above problem will be as follow:



The Multidimensional database are as:

Table	Created From
Time_dim	Created using Order date
Source	Copied from Order_Source
Product	Copied from Product
Product_Type	Copied from Product_Type
Customer	Copied from Customer
Fact_Table	All the keys are copied from transaction table. Total_Revenue is sum of order_amount and Total_Number_Of_Customer is count of customer_id

Deriving management reports from Multidimensional modelling is easy as compared to Entity Relationship modelling therefore it helps for BI Applications like forecasting.

Conclusion

Multidimensional Modelling is preferred when the usage is for BI applications like reporting or decision making while if the application involves storing transaction then Entity Relationship Modelling is preferred.

6. References

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM Sigmod record*, *26*(1), 65-74.

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly*, *36*(4), 1165-1188.

Negash, S. (2004). Business intelligence.

Paim, F. R. S., & de Castro, J. F. B. (2003). *DWARF: An approach for requirements definition and management of data warehouse systems.* Paper presented at the Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International.