Office Use Only

Monash University

Semester Examination Period
Faculty of Information Technology

Sample Exam SOLUTIONS

EXAM CODES: **Sample-1**

TITLE OF PAPER: **Sample-1**

EXAM DURATION:

READING TIME:

THIS PAPER IS FOR STUDENTS STUDYING AT:(tick where applicable)

☐ Berwick	☒ Clayton	☐ Malaysia	☒ Off Campus Learning	☐ Open Learning
☒ Caulfield	☐ Gippsland	☐ Peninsula	☐ Enhancement Studies	☐ Sth Africa
☐ Pharmacy	☐ Other (specify)			

During an exam, you must not have in your possession, a book, notes, paper, calculator, pencil case, mobile phone or other material/item which has not been authorised for the exam or specifically permitted as noted below. Any material or item on your desk, chair or person will be deemed to be in your possession. You are reminded that possession of unauthorised materials in an exam is a discipline offence under Monash Statute 4.1.

No examination papers are to be removed from the room.

Attempt all questions. All answers must be **printed neatly** on this paper. Answer questions with concisely expressed factual information. The backs of pages in this paper may be used for any rough work. **Any material written on the backs of pages will not normally be corrected**. If an answer needs to overflow from its designated answer space to a blank page, clearly indicate that this is the case and that the material on the blank page is for correction. This paper must be handed up at the end of the examination, even if no questions are attempted. **There are 8 questions, each of which is worth 10 marks. The total is 80 marks.** This exam counts as 60% of the final assessment for the unit.

AUTHORISED MATERIALS

CALCULATORS	☐ **YES**	☒ **NO**
OPEN BOOK	☐ **YES**	☒ **NO**
SPECIFICALLY PERMITTED ITEMS	☐ **YES**	☒ **NO**

Candidates must complete this section if required to write answers within this paper

STUDENT ID __ __ __ __ __ __ __ __ __ DESK NUMBER __ __ __ __

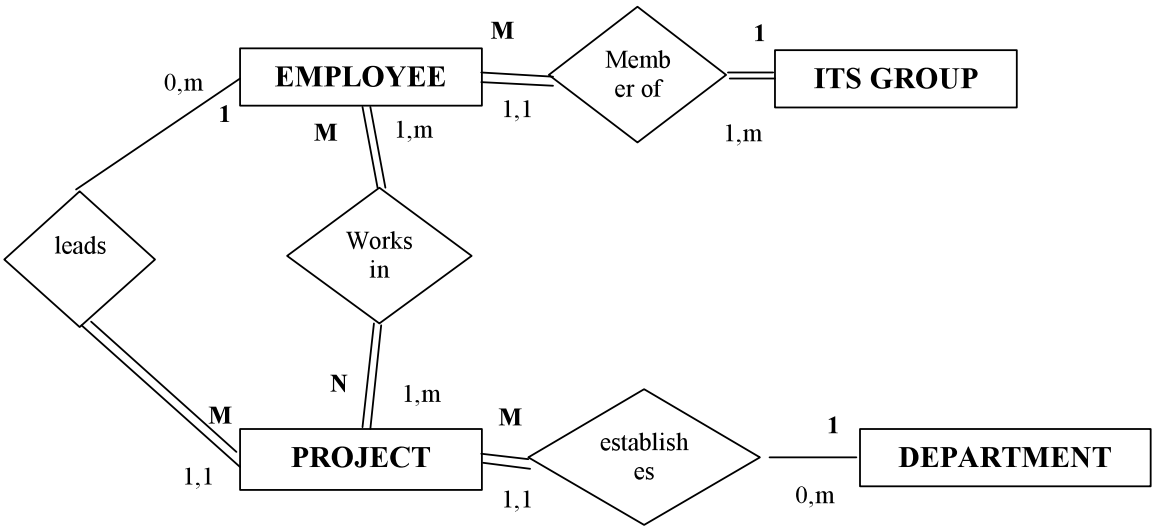
Question 1:

The Information Technology Services (ITS) department at Monash University needs a database to keep track of ITS staff members and the projects they are working on. The system must satisfy the following requirements:

- For each ITS employee, list the employee number, employee name, job title (position), and the number and name of the ITS group in which he or she works. In addition, for each project to which the employee is assigned, list the project number and name, the percent of the employee's time assignment to the project, and the total number of hours the employee has worked on the project so far.

- For each ITS project, list the project number and name (description), the name of the department that requested the project, the name of the contact person in that requesting department, the project type (maintenance, database, etc.), project status, start data, end date, total budgeted person-hours, total budgeted dollars, and the name of the ITS employee serving as project leader.

The E/R diagram for the above process is given as follows:



The system is now operating with the following tables:

EMPLOYEE (**emp#,** empname, position, ITSgroup#*)*
ITS-GROUP (**itsgroup#,** itsgroupname)
DEPARTMENT (**dept#,** deptname)
PROJECT (**proj#,** projname, contactperson, projtype, projstatus, start_date, end_date, budgperhours, budgtotals$, emp#, dept#*)*
EMP-PROJ (**emp#, proj#,** %timeassignment, projemptotalhrs)

Monash University management would like to analyse the performance of the ITS department, and in order to assist this process, you are asked to develop a data warehouse for analysis purposes. The analysis is needed for identifying at least the following questions:

- What is the total number of projects that are long duration?
- What is the total budget for a certain project type?
- What is an average budget cost per hour of a certain department?

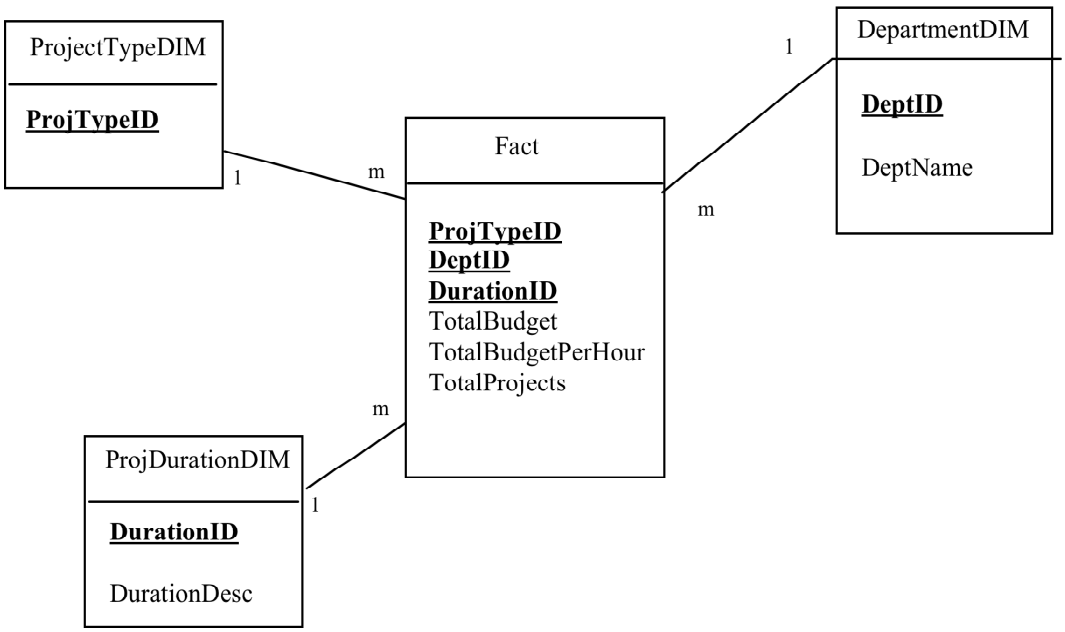
Based on the above requirements, the fact measures that the management is interested in are total budgets, average budget per hours, and total projects; and the dimensions are project type, department, and project duration. Assuming that the management classifies projects into short term (less than 10 days), medium term (between 10 and 30 days), and long term (more than 30 days).

Questions:

a. Draw a star schema containing dimensions and fact, together with their attributes.
b. Define the SQL statements for the implementation of the star schema.
c. Write the SQL statements to answer the above three query requirements.

Write your answer here:

Solution a



Solution b

```
-- create dimension tables
Create Table ProjectTypeDIM As
Select Distinct ProjType as ProjTypeID
From Project;

Create Table DepartmentDIM As
Select Dept# as DeptID, DeptName
From Department;

Create Table ProjDurationDIM
(DurationID   Number;
 DurationDesc Varchar2(20));

Insert Into ProjDurationDIM Values (1, 'Short Duration');
Insert Into ProjDurationDIM Values (2, 'Medium Duration');
Insert Into ProjDurationDIM Values (3, 'Long Duration');
```

Continue your answer here:

```
-- create TempFact
Create Table TempFact As
Select P.ProjType, P.Dept#, P.Start_Date, P.End_Date,
P.budgtotals$, P.Proj#, P.BudgPerHours
From Project P;

Alter Table TempFact
Add (DurationID Number);

Update TempFact
Set DurationID = 1
Where End_Date - Start_Date < 10;

Update TempFact
Set DurationID = 2
Where End_Date - Start_Date >= 10
And End_Date - Start_Date <=30;

Update TempFact
Set DurationID = 3
Where End_Date - Start_Date > 30;

-- create Fact
Create Table Fact As
Select ProjType, Dept#, DurationID,
  SUM(budgtotals$) as TotalBudget,
  SUM(BudgPerHours) as TotalBudgetPerHour,
  COUNT(Proj#)as TotalProjects
From TempFact
Group By ProjType, Dept#, DurationID;
```

Continue your answer here:

Solution c:

- What is the total number of projects that are long duration?

```
Select D.DurationDesc, Sum(TotalProjects)
From Fact F, ProjDurationDIM D
Where F.DurationID = D.DurationID
Group By D.DurationDesc;
```

- What is the total budget for a certain project type?

```
Select T.ProjTypeID, Sum(TotalBudget)
From Fact F, ProjTypeDIM T
Where F.ProjTypeID = T.ProjTypeID
Group By T.ProjTypeID;
```

- What is an average budget cost per hour of a certain department?

```
Select
   D.DeptName,
   Sum(TotalBudgetperHour)/Sum(TotalProjects) as Avg_Budget_Cost
From Fact F, DepartmentDIM D
Where F.DeptID = D.DeptID
Group By D.DeptName;
```

Question 2:

Explain the differences between **`RANK()`**, **`DENSE_RANK()`**, and **`ROW_NUMBER()`** in OLAP SQL.

Write your answer here:

Suppose there is table called some_table, which consists of the following records:

```
create table some_table (
  a number,
  b varchar2(10)
);
```

a	b
5	*
2	*
5	*
3	*
5	*
3	*
2	*
4	#
8	#
4	#
8	#
4	#

```
select dense_rank() over (partition by b order by a) dr,
             rank() over (partition by b order by a) rk,
       row_number() over (partition by b order by a) rn,
       a,b
from some_table;
```

```
        DR         RK         RN          A B
---------- ---------- ---------- ---------- ----------
         1          1          1          4 #
         1          1          2          4 #
         1          1          3          4 #
         2          4          4          8 #
         2          4          5          8 #
         1          1          1          2 *
         1          1          2          2 *
         2          3          3          3 *
         2          3          4          3 *
         3          5          5          5 *
         3          5          6          5 *
         3          5          7          5 *
```

Continue your answers here:

Another example

DR	RK	RN
1	1	1
1	1	2
1	1	3
2	4	4
2	4	5
2	4	6
2	4	7
3	8	8
3	8	9
4	10	10
4	10	11
4	10	12

Notes:
•If the query is to get dense rank <= 5, then all the twelve records will be retrieved.
•If the query is to get rank <=5, then the first seven records will be retrieved.
•If the query is to get row number <=5, then only the first five records will be retrieved.
•Each of the above queries has a different semantic. Additionally, row number has to tie in with a primary key, so that the results will be deterministic. If, for example, the row number is based on $sales, and it happens that the first 10 records have the same $sales figure, then the query for row number <=5 will not always produce the same results.

Question 3:

Monash International would like to analyse their policy in regard to English requirement for admission into a course. Monash International has the following data:

Table: Student_IELTS

Student ID	Student Name	Listening	Reading	Writing	Speaking	Overall
228493	Sooying Tan	6.5	6.5	6.0	7.0	6.5
229094	Xuebing Lu	5.5	5.5	5.5	5.5	5.5
231289	Amandh Kumar	6.0	7.0	6.0	7.0	6.5
234354	Agus Hidayat	5.5	6.0	6.0	6.5	6.0
234355	Budi Rahayu	7.0	7.0	7.0	7.0	7.0
...						
...						

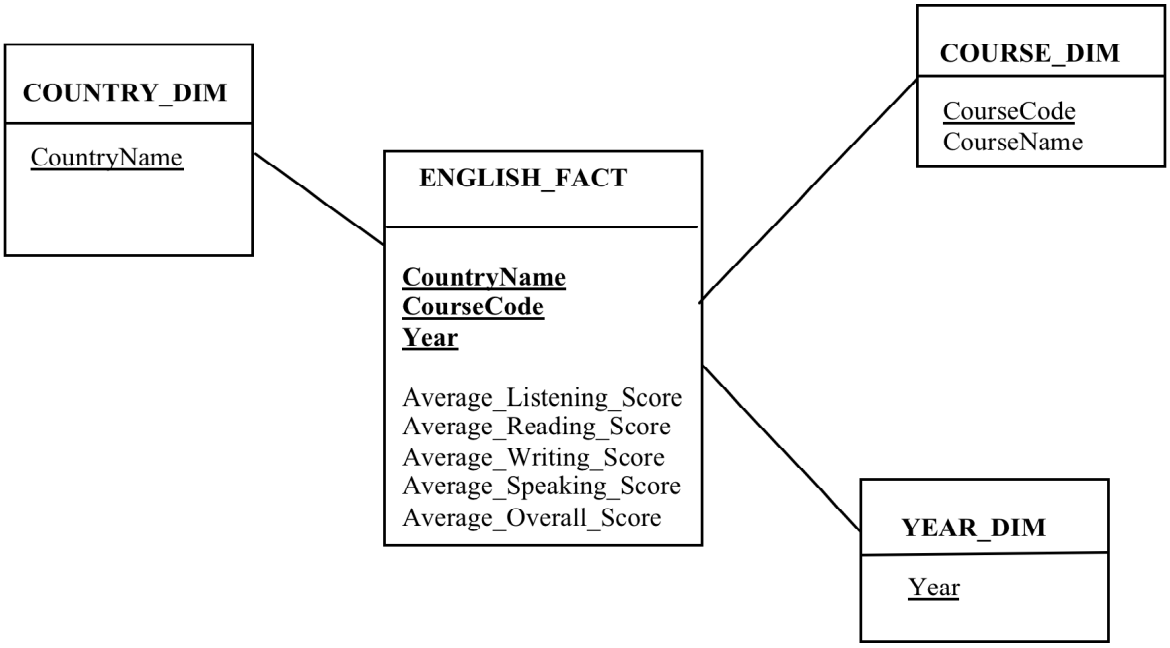
Table: Student_Course

Student ID	Student Name	Course	StarYear
228493	Sooying Tan	MBIS	2013
229094	Xuebing Lu	MBIS	2013
231289	Amandh Kumar	MIT	2013
234354	Agus Hidayat	MIT	2013
234355	Budi Rahayu	MIT	2013
...			
...			

Table: Student

Student ID	Student Name	Address	Suburb	Phone Number	Country
228493	Sooying Tan				Singapore
229094	Xuebing Lu				China
231289	Amandh Kumar				India
234354	Agus Hidayat				Indonesia
234355	Budi Rahayu				Indonesia
...					
...					

A data warehouse based on the above data has been created. A star schema is shown as follows:



Questions:

a. The above star schema will not produce a correct analysis of the fact measures. Explain why. Explain your answer using more concrete examples or data.

b. How do you correct this problem by changing the fact measures of the above star schema. Explain your solution using more concrete examples or data.

Write your answer here:

Question-a

If the operational database contains 10 students from Indonesia doing MIT in 2013, the fact table will contain only one entry, which aggregates these 10 students. The fact will aggregate the IELTS scores (i.e. listening, reading, writing, speaking, and overall score) and get the average (e.g. average listening, average reading, average writing, average speaking, and average overall score). This will be done for each of combination of country, course, and year.

However, if in the OLAP, we would like to get the average scores of students from a specific country, say Indonesia, then we will need to get all the records where country is Indonesia from the fact table. And then average it again.

The records in the fact table are already averaged. If we average on top of another average, the result will be incorrect.

Continue your answer here:

For example, the fact table of the above sample data is shown as follows:

Table: Fact

CountryName	CourseCode	Year	AvgList	AvgRead	AvgWrit	AvgSpeak	AvgOvrall
China	MBIS	2013	5.5	5.5	5.5	5.5	5.5
India	MIT	2013	6.0	7.0	6.0	7.0	6.5
Indonesia	MIT	2013	6.25	6.5	6.5	6.75	6.5
Singapore	MBIS	2013	6.5	6.5	6.0	7.0	6.5

If we query the **Average Listening Score of MIT** students, then based on the above fact table, the average listening score is **(6.0+6.25)/2 = 6.125**.

Supposed in the operational database, there are three students from India and Indonesia (e.g. Amandh from India, and Agus and Budi from Indonesia):

Table: Student_IELTS

Student ID	Student Name	Listening	Reading	Writing	Speaking	Overall
231289	Amandh Kumar	6.0	7.0	6.0	7.0	6.5
234354	Agus Hidayat	5.5	6.0	6.0	6.5	6.0
234355	Budi Rahayu	7.0	7.0	7.0	7.0	7.0

If we look at these three students from India and Indonesia (Amand, Agus, and Budi), the average listening score is **(6.0+5.5+7.0)/3 = 6.17**.

This example shows that keeping the average score in the fact will produce incorrect OLAP query results.

Question-b.
Hence, average is not a good fact measures. We should not have average_reading, average_listening, etc in the fact.

On the other hand, we should have total_reading, total_listening, total_speaking, total_writing, and total_overall in the fact. We also need total_number_of_students in the fact.

So, in OLAP, if we want to get the **Average Listening Score of MIT** course students, we will get the **total_listening** and **total_number_of_students**, and then divide them to get the average. Using the example above, **total_listening = (6.0+5.5+7.0) = 18.5**, and **total_number_of_students = 3**. Then **listening average is 18.5/3 = 6.17**

Question 4:

There is a tool way (or tool road) in a metropolitan city (such as CityLink or EastLink in Melbourne, or any similar tool roads in other major cities in the world). This tool way has a number of gates, where the motorist needs to pay. Every time a motorist passes through this tool gate, the registration number of the vehicle, vehicle type (e.g. car, bus, truck, etc), amount paid, and time, are recorded in the operational database.

A data warehouse needs to be built, for analysing the *revenue* from the toll payments. The management would like to drill down this revenue based on the *tollgate* (there is a number of toll gates along the toll way), *day of week* (e.g. weekdays, weekends), and *time period of a day* (e.g. peak hours, non-peak hours, late nights).

You are required to draw **three levels of star schemas** showing three different levels of aggregation for the above data warehouse. You also need to explain each of the three star schemas, by contrasting the level of aggregation. Level-0 star schema contains the most detailed data, whereas level-2 star schema is the highly aggregated (e.g. containing highly aggregated data).

Questions:

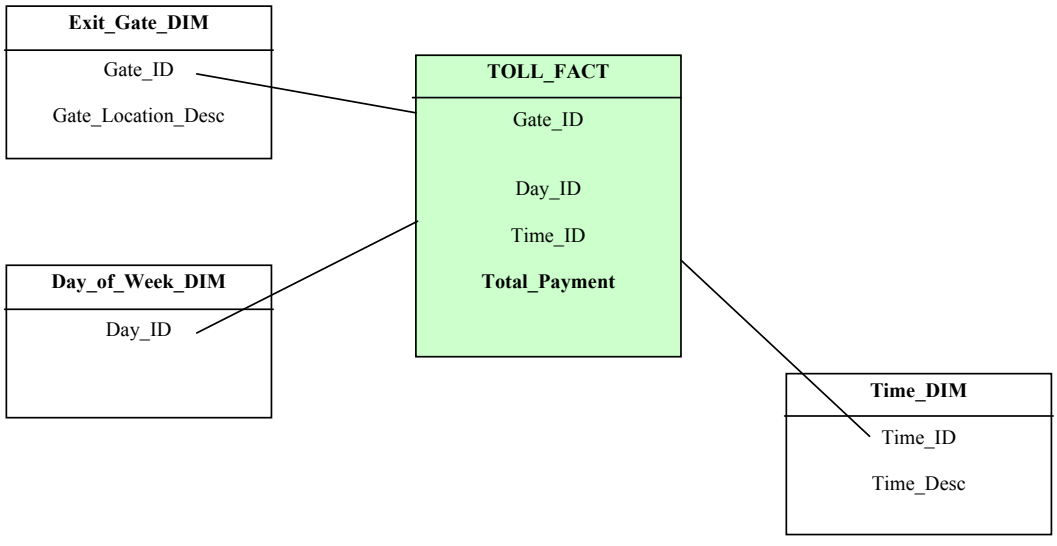
(a) Draw a level-2 star schema and explanation why it is a level-2 schema

(b) Draw a level-1 star schema and explain why it is a level-1 schema. You may want to add a new dimension, called *vehicle* (e.g. cars, trucks, busses, etc). You need to also explain the difference between level-1 and level-2 schemas.

(c) Draw a level-0 star schema and explain why it is a level-0 schema. You also need to explain the difference between level-1 and level 0 schemas.

Write your answer here:

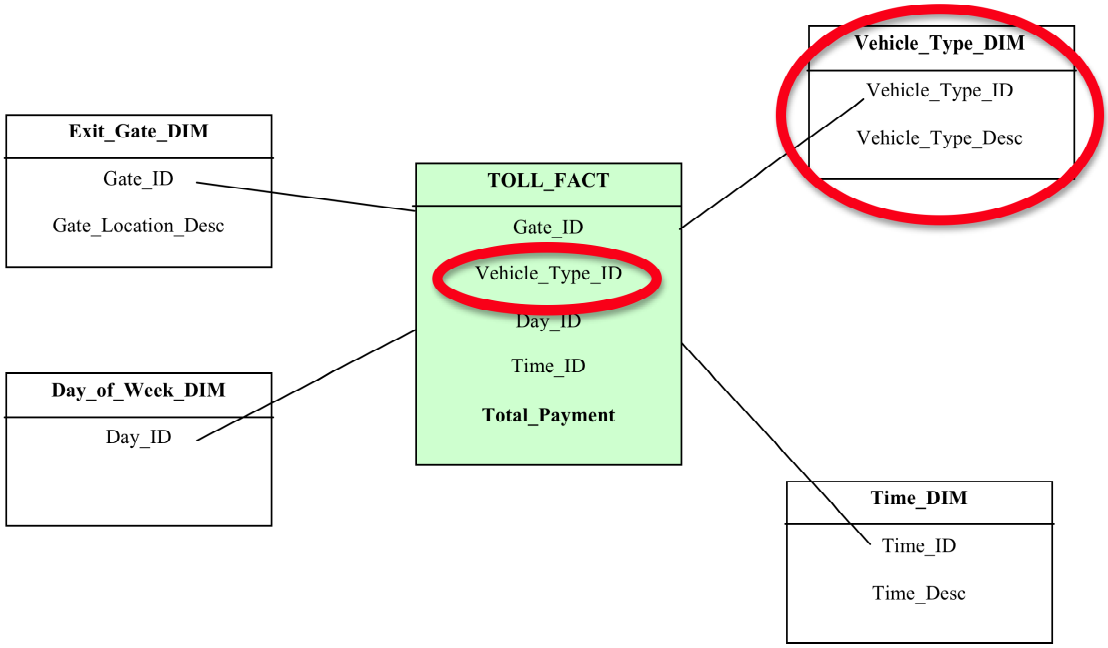
Level 2 – Highly aggregated. This is the most highly aggregated data where we keep "Total Payment" in the Fact table, and the Dimensions are based on Days of Week (not the actual date), and Time Period (not the actual travel time).



Continue your answer here:

Level 1– We add with an additional dimension, for example the Vehicle Type Dimension. In the Vehicle Type Dimension, we store the vehicle types, such as bus, truck, cars, etc. Note that we do not keep the "Registration Number" of the car. The granularity is at the vehicle type level, not at an individual car level.

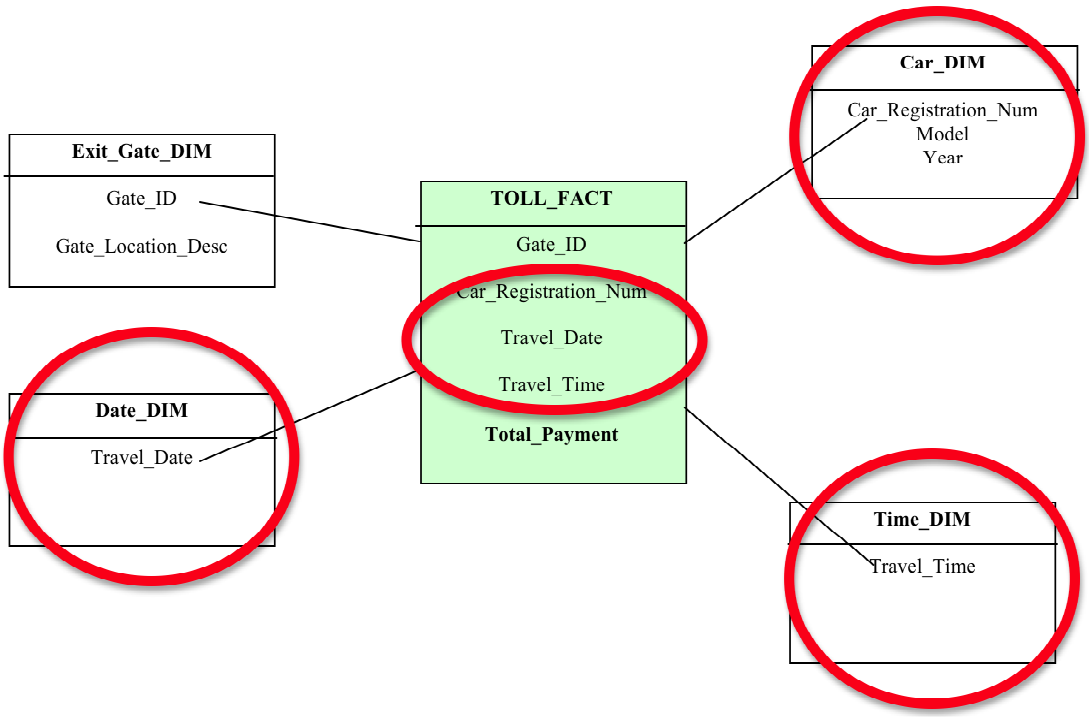
So, from level 2 to level 1, the total payment in level 2 is broken down by Vehicle Type. Other dimensions remain unchanged.



Continue your answer here:

Level 0– Detail Level

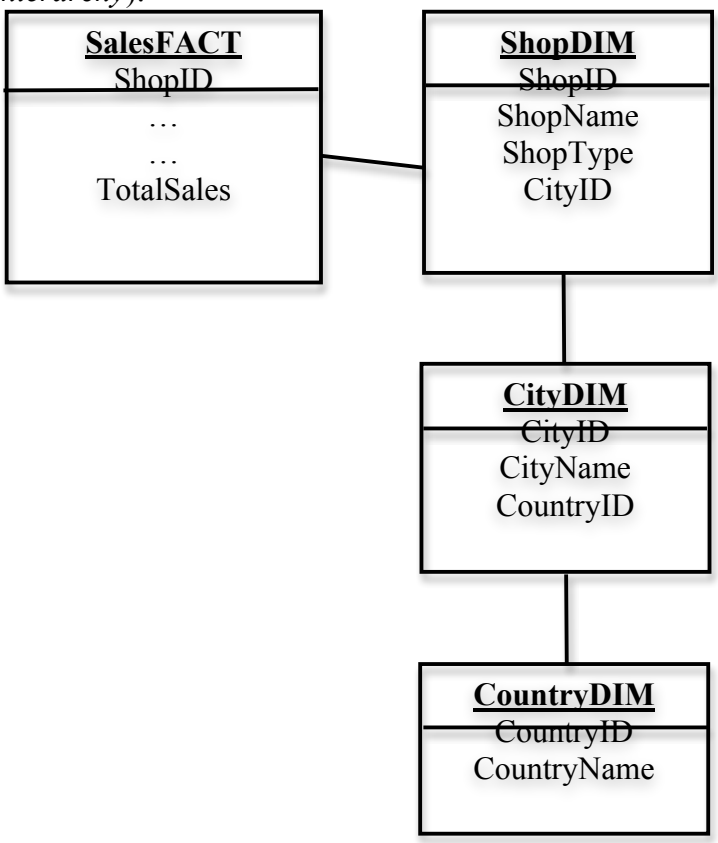
This is the most detail level as we have Date_DIM and Time_DIM to indicate the actual travel dates and travel time; as well as Car DIM to indicate the actual car (rather than just the type of vehicle)



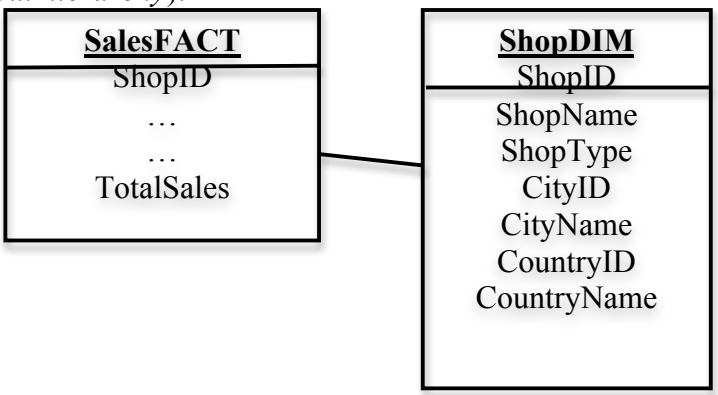
Question 5:

Consider the following star schemas. Star Schema-1 contains a hierarchy in the dimension, whereas Star Schema-2 collapses the hierarchy into one dimension.

Star Schema-1 (*with hierarchy*):



Star Schema-2 (*without hierarchy*):



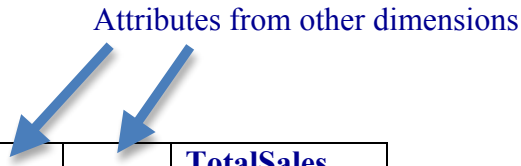
Questions:

(a) Draw sample table contents of the fact and dimension tables of the two star schemas.

(b) Compare and contrast the two star schemas using the sample tables in question (*a*) above. Explain the pros and cons of each star schema.

Write your answers here:

(a)
Star Schema-1



ShopFACT

ShopID			**TotalSales**
1	…	…	$1,500,000
2	…	…	$2,750,000
3	…	…	$1,800,000

ShopDIM

ShopID	**ShopName**	**ShopType**	**CityID**
1	Myer	DepartmentStore	C
2	Coles	GrocerryStore	C
3	BigW	DepartmentStore	M

CityDIM

CityID	**CityName**	**CountryID**
C	Canberra	AD
M	Melbourne	AD

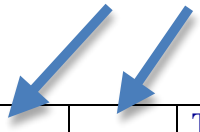
CountryDIM

CountryID	**CountryName**
AD	Australia

Continue your answers here:

Star Schema-2

Attributes from other dimensions



ShopFACT

ShopID			**TotalSales**
1	…	…	$1,500,000
2	…	…	$2,750,000
3	…	…	$1,800,000

ShopDIM

ShopID	**ShopName**	**ShopType**	**CityID**	**CityName**	**CountryID**	**CountryName**
1	Myer	DepartmentStore	C	Canberra	AD	Australia
2	Coles	GrocerryStore	C	Canberra	AD	Australia
3	BigW	DepartmentStore	M	Melbourne	AD	Australia

(b)

Star Schema-1

Pros: normalized, minimized data duplication

Cons: When producing a report, we need to join the fact table with three dimension tables: shopDim, cityDim, and countryDim. Hence, we need three join operations.

Star Schema-2

Cons: unnormalized, has data duplication

Pros: When producing a report, we need to join the fact table with shopDim only. Hence, we need one join operation only.

Question 6:

Data cleaning is an important part in building a clean and correct data warehouse. Data cleaning is often needed, because there are mistakes and inconsistencies in the operational database. Before data cleaning is done, we need to do data exploration on the operational database in order to find out if there are any mistakes and inconsistencies in the operational database.

The following are the four tables in the operational database:

```
SQL> desc dw.uselog;
 Name                                      Null?    Type
 ----------------------------------------- -------- -----------------
 LOG_DATE                                  NOT NULL DATE
 LOG_TIME                                  NOT NULL DATE
 STUDENT_ID                                NOT NULL CHAR(11)
 ACT                                                CHAR(1)

SQL> desc dw.student;
 Name                                      Null?    Type
 ----------------------------------------- -------- -----------------
 SEX                                                VARCHAR2(2)
 FULL_PART                                          VARCHAR2(2)
 TYPE                                               VARCHAR2(4)
 CLASS_ID                                           VARCHAR2(6)
 MAJOR_CODE                                         VARCHAR2(8)
 STUDENT_ID                                NOT NULL CHAR(11)

SQL> desc dw.class;
 Name                                      Null?    Type
 ----------------------------------------- -------- -----------------
 CLASS_DESCRIPTION                                  CHAR(50)
 CLASS_ID                                           VARCHAR2(6)

SQL> desc dw.major;
 Name                                      Null?    Type
 ----------------------------------------- -------- -----------------
 MAJOR_NAME                                         CHAR(35)
 MAJOR_CODE                                         VARCHAR2(8)
```

Questions:

(a) Write the SQL command to find out if there are duplicate student records

(b) Write the SQL command to find out if there are records in dw.uselog whereby the Student_ID exists in dw.uselog actually do not exist in dw.student

Write your answers here:

(a)

```
select student_id, count(*)
from dw.student
group by student_id
having count(*) > 1;
```

Notes for studying: students need to understand the difference between WHERE and HAVING

(b)

```
select *
from dw.uselog
where student_id NOT IN
  (select student_id from dw.student);
```

Notes for studying: students need to understand nested queries

Question 7:

An established real estate agent in Melbourne has started their business many years ago and has implemented a very simple database system. The simple database system consists of *one* large table listed below.

Table Name: PROPERTY	
Field Name	Description
Key	Unique key
Date_offered	Date property offered to the public
Summary	Short description of the property
Adtext	Longer description of the property
Url	The URL of the advertisement
Address	Property address
Suburb	Property suburb name
Postcode	Property postcode
Longitude	Longitude of address
Latitude	Latitude of address
Category	'Residential' or 'Commercial'
Zoning	Commercial Zoning Type
Property_type	Residential Property Type: 'House', 'apartment', or 'lot'
Houseprice	Price of property
Num_bedrooms	Number of bedrooms
Lot_size	Size of the lot
Heating	'ducted', 'gas', 'open fireplace' or 'wood'
Garage	Type of garage
Ensuite	'yes' or 'no'
Balcony	'yes' or 'no'
Pool	'yes', 'no'
Tennis_court	'yes', 'no'
Spa	'yes', 'no'
Aspect_facing	'north', 'south', 'east', or 'west'
School_distance	Distance to nearest school – in km
Shops_distance	Distance to nearest shops – in km
Train_distance	Distance to nearest train station – in km
Bus_distance	Distance to nearest bus stop – in km
Hospital_distance	Distance to nearest hospital – in km
Major_road_distance	Distance to nearest major road – in km

Based on this information, the manager of the real estate agent requires a data warehouse for analysis purposes. The final star schema may include a junk dimension.

Questions:

a. What is a junk dimension? Explain!!! Also, use a sample data to illustrate a junk dimension

b. Design two versions of star schema for the above case study; one without a junk dimension, and the other with a junk dimension. Compare and contrast these two schemas; focusing on the junk dimension only

Write your answer here:

Solution a:

A junk dimension is a dimension where each of the non-key attributes contains only a small number of possible values (e.g. Yes/No, North/South/West/East, 1/2/3). Therefore the content of a junk dimension is a Cartesian product of the values of all its attributes.

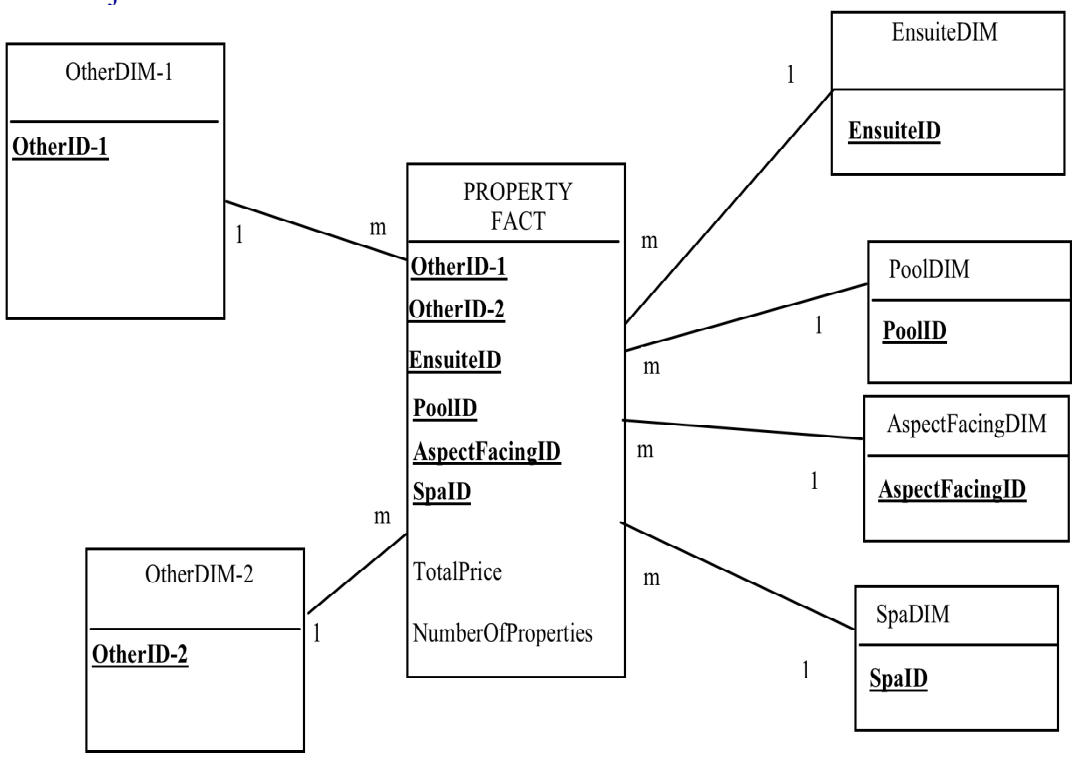
Sample data of a junk dimension:

JunkID	**EnsuiteID**	**PoolID**	**AspectFacingID**	**SpaID**
1	Yes	Yes	North	Yes
2	Yes	Yes	North	No
3	Yes	Yes	South	Yes
4	Yes	Yes	South	No
5	Yes	Yes	East	Yes
6	Yes	Yes	East	No
7	Yes	Yes	West	Yes
8	Yes	Yes	West	No
9	Yes	No	North	Yes
10	Yes	No	North	No
11	Yes	No	South	Yes
12	Yes	No	South	No
13	Yes	No	East	Yes
14	Yes	No	East	No
15	Yes	No	West	Yes
16	Yes	No	West	No
17	(repeat the above for **No** Ensuite)			
	…			
	…			
32	…			

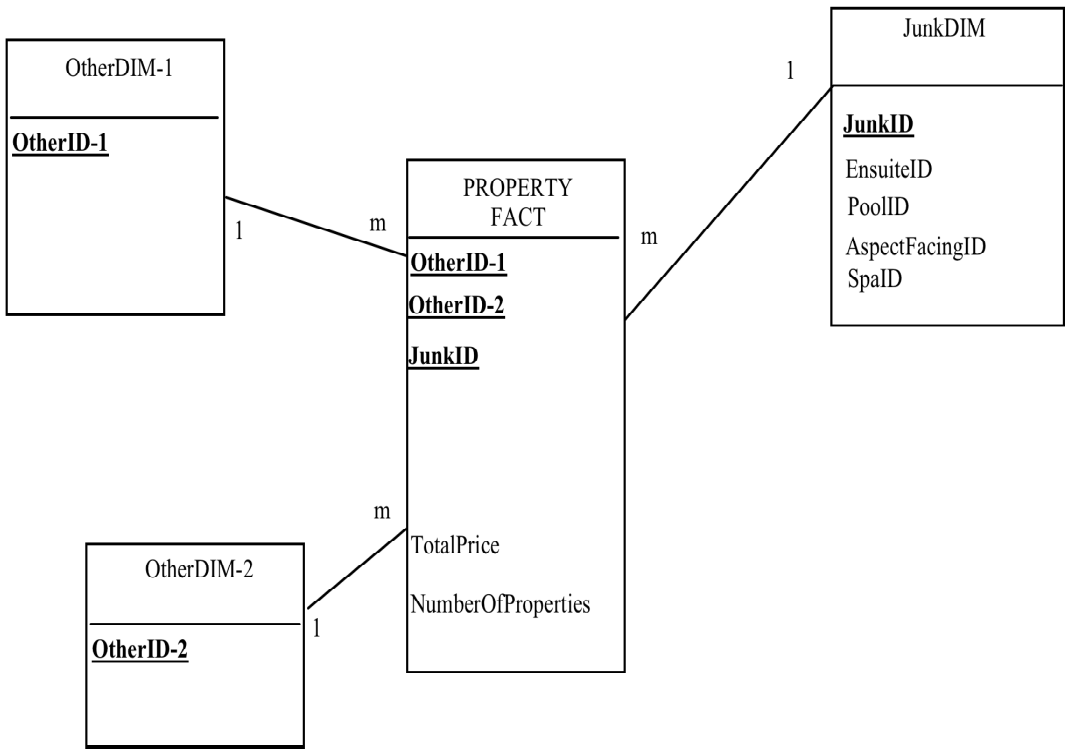
Continue your answer here:

Solution b:

Non-junk dimension star schema



Junk dimension star schema



The junk dimension basically combines (through a Cartesian product) all the small dimensions into one junk dimension.

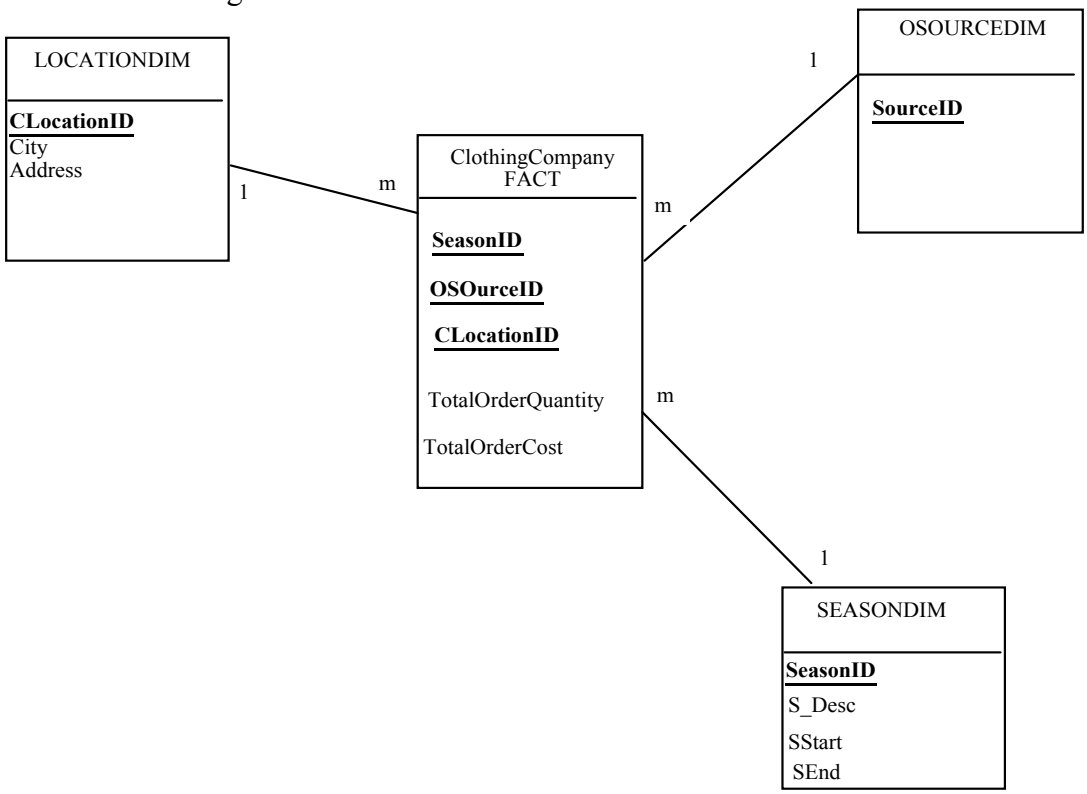
Continue your answer here:

Non-junk:
- If we need to create a report involving these four dimensions, we need to have three join operations.
- If we need to have one additional record to the dimension (e.g. aspect facing dimension has one more record called NorthEast, for example), then maintaining the dimension is rather straightforward.

Junk:
- We only need to join the fact and the junkdim. Having one join operation is always better than three join operations.
- If we need to add an additional record to the junk, it will be more difficult.
- Junk dimension is limited to handling small dimensions only. Larger dimensions are not advised to be included in the junk.
- Junk dimension is only suitable to handle a few small cardinality dimensions.

Question 8:

Given the following schema:



The tables (e.g. ClothingCompany fact and the three dimensions) have been created and populated with an adequate number of records.

The table names and attributes are shown in the above star schema. In the Fact table, the total order quantity and total order cost attributes are included.

Write the SQL for the following advanced OLAP queries:

a) Perform a **CUBE** operation (use all dimensions). Display each TotalOrderCost and the subtotals.
b) Like question (a) above, but now perform a **ROLLUP** operation.
c) Perform a **CUMMULATIVE SUM** of the TotalOrderCost of all WEBSITE orders (use all dimensions).
d) Like question (c) above, perform a CUMMULATIVE SUM of the TotalOrderCost but **PARTITIONED** based on the OSourceID, that is one partition for Phone orders, one partition for Fax orders, and one partition for Website orders.
e) Show the total order costs of each source order, and **RANK** them.
f) Display the source order that generates the highest total order cost.

Write your answer here:

a)

```
Select l.city, s.S_Desc, o.sourceid, sum(c.TotalOrderCost) as OrderCost
from LocationDim l, SeasonDim s, OsourceDim o, ClothingCompanyFact c
where l.CLocationID = c.CLocationID
and s.SeasonID = c.SeasonID
and o.sourceid = c.sourceID
group by cube (l.city, s.S_Desc, o.sourceid);
```

Notes:

1. In the above SQL, assume we use l.city (from LocationDIM), and s.s_desc (from SeasonDIM)
2. Alternatively, because OSourceDIM has only one attribute sourceID, we can use sourceID attribute in the fact. In the real-life, OSourceDIM has many more attributes, and therefore, a join with the OSourceDIM table is necessary
3. You can use DECODE and GROUPING for better formatting

```
Select l.city, s.S_Desc, c.sourceid, sum(c.TotalOrderCost) as OrderCost
from LocationDim l, SeasonDim s, ClothingCompanyFact c
where l.CLocationID = c.CLocationID
and s.SeasonID = c.SeasonID
group by cube (l.city, s.S_Desc, c.sourceid);
```

b)

```
same as (a), but change the cube with rollup
```

Continue your answer here:

```
c)
Select l.city, s.S_Desc, sum(c.TotalOrderCost),
   TO_CHAR(SUM(SUM(c.TotalOrderCost))
   OVER(ORDER BY l.city, s.S_Desc ROWS UNBOUNDED PRECEDING),
   '9,999,999.99') AS Cummulative_Total_Order_Cost
from LocationDim l, SeasonDim s, OsourceDim o, ClothingCompanyFact c
where l.CLocationID = c.CLocationID
and s.SeasonID = c.SeasonID
and o.sourceid = c.sourceID
and o.SourceID = 'WEBSITE'
group by (l.city, s.S_Desc);
```

Notes:

1. The above will produce a listing sorted by City and then by Season (e.g. seasons within the same city will be sorted according to the season). The cumulative will be based on this ordering. This will only work if the query has ONE city. If it has multiple cities, it will not make sense that the cumulative total order cost for the second city will start from the first city the first season. Hence, the above query should be like this (e.g. to limit to one city only, such as Melbourne):

```
Select l.city, s.S_Desc, sum(c.TotalOrderCost),
   TO_CHAR(SUM(SUM(c.TotalOrderCost))
   OVER(ORDER BY l.city, s.S_Desc ROWS UNBOUNDED PRECEDING),
   '9,999,999.99') AS Cummulative_Total_Order_Cost
from LocationDim l, SeasonDim s, OsourceDim o, ClothingCompanyFact c
where l.CLocationID = c.CLocationID
and s.SeasonID = c.SeasonID
and o.sourceid = c.sourceID
and o.SourceID = 'WEBSITE'
and l.city = 'MELBOURNE'
group by (l.city, s.S_Desc);
```

2. If we would like to include multiple cities in the report, the cumulative needs to have a PARTITION, based on city. Hence, for each city, there will be a separate cumulative:

```
Select l.city, s.S_Desc, sum(c.TotalOrderCost),
   TO_CHAR(SUM(SUM(c.TotalOrderCost))
   OVER(PARTITION BY l.city
   ORDER BY l.city, s.S_Desc ROWS UNBOUNDED PRECEDING),
   '9,999,999.99') AS Cummulative_Total_Order_Cost
from LocationDim l, SeasonDim s, OsourceDim o, ClothingCompanyFact c
where l.CLocationID = c.CLocationID
and s.SeasonID = c.SeasonID
and o.sourceid = c.sourceID
and o.SourceID = 'WEBSITE'
group by (l.city, s.S_Desc);
```

Continue your answer here:

d) Note that we assume that we limit the query to ONE city only:

```
Select o.SourceID, l.city, s.S_Desc, sum(c.TotalOrderCost),
   TO_CHAR(SUM(SUM(c.TotalOrderCost))
   OVER(PARTITION BY o.SourceID
   ORDER BY o.SourceID, l.city, s.S_Desc ROWS UNBOUNDED PRECEDING),
   '9,999,999.99') AS Cummulative_Total_Order_Cost
from LocationDim l, SeasonDim s, OsourceDim o, ClothingCompanyFact c
where l.CLocationID = c.CLocationID
and s.SeasonID = c.SeasonID
and o.sourceid = c.sourceID
and l.city = 'MELBOURNE'
group by (O.SourceID, l.city, s.S_Desc);
```

Notes:

1. The above will produce a listing sorted by SourceID, and then City and then by Season. In this case, there is only one City, which is Melbourne.
2. Each SourceID will have it's own cumulative.
3. If you have multiple cities, you could have a separate cumulative for SourceID combined with City. It means SourceID 'Phone' and City 'Melbourne' will have one set of cumulative, whereas SourceID 'Phone' and City 'Sydney' will have a separate set of cumulative:

```
Select o.SourceID, l.city, s.S_Desc, sum(c.TotalOrderCost),
   TO_CHAR(SUM(SUM(c.TotalOrderCost))
   OVER(PARTITION BY o.SourceID, l.City
   ORDER BY o.SourceID, l.city, s.S_Desc ROWS UNBOUNDED PRECEDING),
   '9,999,999.99') AS Cummulative_Total_Order_Cost
from LocationDim l, SeasonDim s, OsourceDim o, ClothingCompanyFact c
where l.CLocationID = c.CLocationID
and s.SeasonID = c.SeasonID
and o.sourceid = c.sourceID
group by (O.SourceID, l.city, s.S_Desc);
```

Continue your answer here:

```
e)
select OsourceID,
       sum(TotalOrderCost)as OrderCost,
       rank() OVER(ORDER BY sum(TotalOrderCost) DESC)
from ClothingCompanyFact c
group by OsourceID;


f)
SELECT *
FROM (
      select OsourceID, sum(TotalOrderCost)as OrderCost,
      rank() OVER(ORDER BY sum(TotalOrderCost) DESC) as orderrank
      from ClothingCompanyFact c
      group by OsourceID;
      )
WHERE orderrank <=1;
```

Notes:

1. Using MAX will be incorrect:

```
select OsourceID, max(TotalOrderCost)
from ClothingCompanyFact;
```

As you cannot mix an attribute (e.g. OSourceID) and an aggregation function (e.g. max) in one Select statement.

2. You can do this, but it will only show the maximum TotalOrderCost without the OSourceID:

```
select max(TotalOrderCost)
from ClothingCompanyFact;
```

3. You cannot do a group by either:

```
select OsourceID, max(TotalOrderCost)
from ClothingCompanyFact
group by OSourceID;
```

because it will get the maximum TotalOrderCost for each sourceID, which is not what the query asks.

THE END