FIT3003 - Assignment 1

Matt Small, Paarth Bhasin, Jacky Yuan

About

Coversheet

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
26356104	*Bhasin*	**Paarth**
25959816	*Small*	**Matthew**
25179578	*Yuan*	**Jacky Chiu Kit**

* Please include the names of all other group members.

Unit name and code	Business Intelligence and Data Warehousing FIT3003	
Title of assignment	FIT3003 Group Assignment	
Lecturer/tutor	Agnes Haryanto	
Tutorial day and time	Wed/Thurs 8am	**Campus** Clayton
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 Sunday 15 October 11:55pm		**Date submitted** Sunday 15 October

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

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

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

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

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 The material used can be from any source (staff, students or the internet, published and unpublished works).

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

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

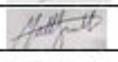
Student Statement:

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

Signature .....*as follows*..... *Date* .....*15/10/2017*.....

*delete (iii) if not applicable

Signature  Date: 15/10/2017 Signature _____ Date: _____

Signature  Date: 15/10/17 Signature _____ Date: _____

Signature  Date: 15/10/17 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

Contribution Declaration Form:

Percentage of Contribution:

Name	Student ID	Contribution
Paarth Bhasin	26356104	33.33%
Matthew Small	25959816	33.33%
Jacky Yuan	25179578	33.33%

List of parts that each person did:

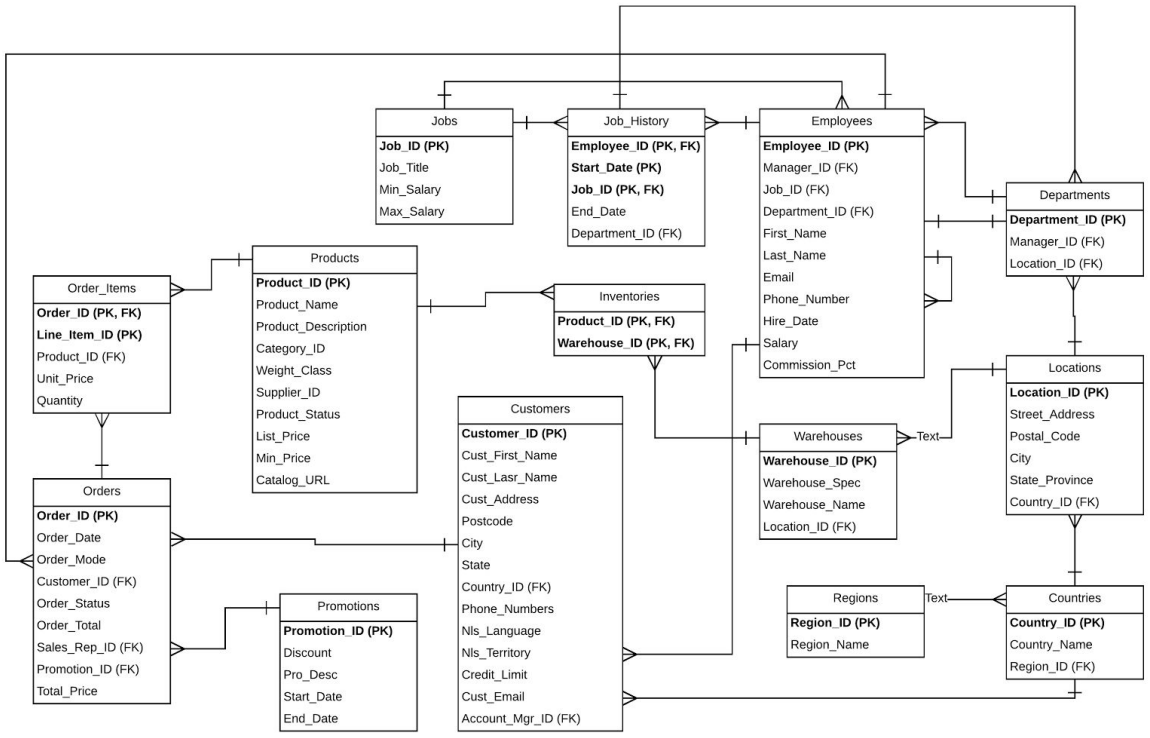
Name	Parts and Questions
Paarth Bhasin	C1, C2, C4 (Reports 1,2,3,4,7,8), C5
Matthew Small	C1, C2, C4
Jacky Yuan	C1, C2, C3

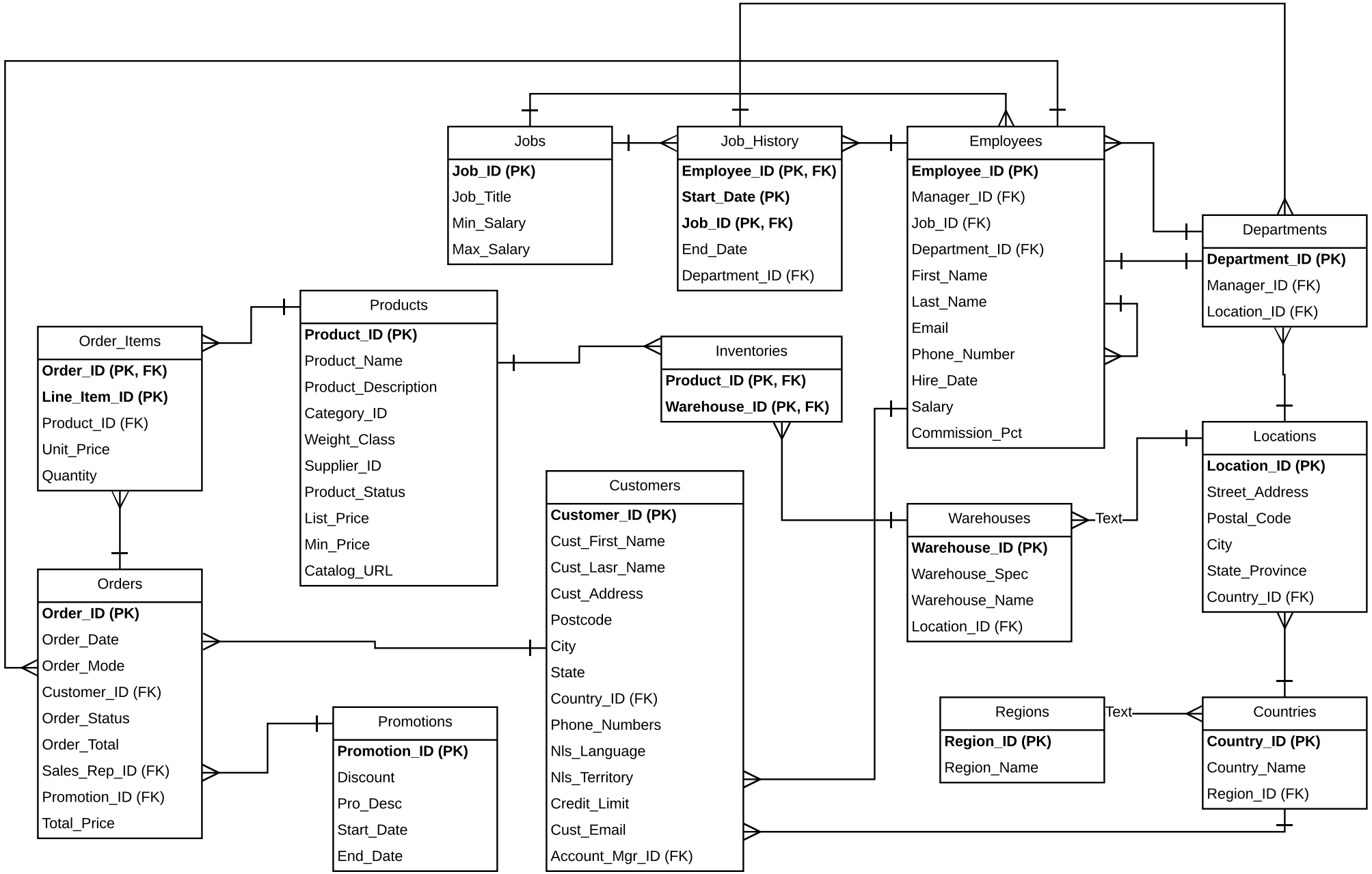
Details of Oracle Accounts:

Connection Name: FIT3003
Username: 26356104
Password: student

Task 1 (PDF on Next Page)

a) Entity Relationship Diagram:




Jobs
Job_ID (PK)
Job_Title
Min_Salary
Max_Salary
Job_History
Employee_ID (PK, FK)
Start_Date (PK)
Job_ID (PK, FK)
End_Date
Department_ID (FK)
Employees
Employee_ID (PK)
Manager_ID (FK)
Job_ID (FK)
Department_ID (FK)
First_Name
Last_Name
Email
Phone_Number
Hire_Date
Salary
Commission_Pct
Departments
Department_ID (PK)
Manager_ID (FK)
Location_ID (FK)
Products
Product_ID (PK)
Product_Name
Product_Description
Category_ID
Weight_Class
Supplier_ID
Product_Status
List_Price
Min_Price
Catalog_URL
Order_Items
Order_ID (PK, FK)
Line_Item_ID (PK)
Product_ID (FK)
Unit_Price
Quantity
Inventories
Product_ID (PK, FK)
Warehouse_ID (PK, FK)
Customers
Customer_ID (PK)
Cust_First_Name
Cust_Lasr_Name
Cust_Address
Postcode
City
State
Country_ID (FK)
Phone_Numbers
Nls_Language
Nls_Territory
Credit_Limit
Cust_Email
Account_Mgr_ID (FK)
Warehouses
Warehouse_ID (PK)
Warehouse_Spec
Warehouse_Name
Location_ID (FK)
Text
Locations
Location_ID (PK)
Street_Address
Postal_Code
City
State_Province
Country_ID (FK)
Orders
Order_ID (PK)
Order_Date
Order_Mode
Customer_ID (FK)
Order_Status
Order_Total
Sales_Rep_ID (FK)
Promotion_ID (FK)
Total_Price
Promotions
Promotion_ID (PK)
Discount
Pro_Desc
Start_Date
End_Date
Regions
Region_ID (PK)
Region_Name
Text
Countries
Country_ID (PK)
Country_Name
Region_ID (FK)

b) Data Cleaning:

1.

Detecting (A duplicate customer exists)

```
select customer_id, count(*)
from HOSALES.CUSTOMERS
group by customer_id
having count(*) > 1;
```

	CUSTOMER_ID	COUNT(*)
1	101	2

Correcting (Remove duplicate customer)

```
Create table Customer as
select h.*, ROW_NUMBER() OVER (Partition By customer_id order by customer_id) as
RowNumber
from HOSALES.CUSTOMERS h
order by customer_id;

Delete from Customer
Where RowNumber <> 1;

Alter Table Customer
Drop Column RowNumber;
```

Before Cleaning:

	CUSTOMER_ID
1	101
2	101
3	102
4	103
5	104
6	105
7	106

After Cleaning:

	CUSTOMER_ID
1	101
2	102
3	103
4	104
5	105
6	106
7	107

2.

Detecting (Duplicate order ID 2359)

```
select order_id, count(*)
from HOSALES.ORDERS
group by order_id
having count(*) > 1;
```

	ORDER_ID	COUNT(*)
1	2359	2

Correcting (Break the common order_id by making one equal to max order Id + 1)

Both the orders seem to be valid orders. So we can't just delete them. We need to break the duplication by assigning one of them a different order ID.

```
Create table Orders as
select o.*, ROW_NUMBER() OVER (Partition By order_id order by customer_id) as RowNumber
from HOSALES.ORDERS o
order by order_id;

Update Orders
Set Order_ID = (Select max(Order_ID)+1 from Orders)
Where RowNumber <> 1;

Alter Table Customer
Drop Column RowNumber;
```

Before Cleaning:

	ORDER_ID
1	2354
2	2355
3	2356
4	2357
5	2358
6	2359
7	2359
8	2360
9	2361
10	2362

After Cleaning:

	ORDER_ID
1	2354
2	2355
3	2356
4	2357
5	2358
6	2359
7	2463
8	2360
9	2361
10	2362

3.

Detecting (Contains an order not in Orders)

```
select * from HOSALES.ORDER_ITEMS h
where h.ORDER_ID NOT IN
(select order_id from HOSALES.ORDERS);
```

	ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	UNIT_PRICE	QUANTITY
1	2459	1	2350	2341.9	6

Correcting

This will not affect our data warehouse because we join on the order and order_item tables. So any non existent orders will not be counted. But if it should be counted as an order then I am not sure what to do.

Before Cleaning: *Stays the same*
After Cleaning: *Stays the same*

4.

Detecting (Contains direct orders that don't have sales representative)

```
select * from HOSALES.ORDERS
where ORDER_MODE = 'direct'
and SALES_REP_ID is NULL
order by ORDER_ID;
```

Correcting (Remove the order from table - hosales.orders not modifiable)

```
Delete from Orders
Where ORDER_MODE = 'direct'
and SALES_REP_ID is NULL;
```

Before Cleaning:

	ORDER_ID	ORDER_DATE	ORDER_MODE	CUSTOMER_ID	ORDER_STATUS	ORDER_TOTAL	SALES_REP_ID	PROMOTION_ID	TOTAL_PRICE
1	2421	12/MAR/07 08:53:54.562432000 PM	direct	109	1	71249	(null)	1	71249
2	2426	17/NOV/06 01:22:11.262552000 AM	direct	148	6	7164	(null)	1	7164
3	2428	10/NOV/07 03:41:34.463567000 AM	direct	116	8	14299.6	(null)	3	12869.64

After Cleaning:

Rows above were deleted.

ORDER_ID	ORDER_D...	ORDER_M...	CUSTOME...	ORDER_S...	ORDER_T...	SALES_RE...	PROMOTI...	TOTAL_P...	ROWNUM...

5.

Detecting (All fields in HOSALES.WAREHOUSES, WAREHOUSE_SPEC are null)

```
Select warehouse_spec
From hosales.warehouses;
```

Correcting (Remove column until information provided)

```
Create table warehouses as
Select *
From hosales.warehouses;

Alter table warehouses
Drop column warehouse_spec;
```

Before Cleaning:

	WAREHOUSE_ID	WAREHOUSE_SPEC	WAREHOUSE_NAME	LOCATION_ID
1	1	(null)	Southlake, Texas	1400
2	2	(null)	San Francisco	1500
3	3	(null)	New Jersey	1600
4	4	(null)	Seattle, Washington	1700
5	5	(null)	Toronto	1800
6	6	(null)	Sydney	2200
7	7	(null)	Mexico City	3200
8	8	(null)	Beijing	2000
9	9	(null)	Bombay	2100

After Cleaning:

	WAREHOUSE_ID	WAREHOUSE_NAME	LOCATION_ID
1	1	Southlake, Texas	1400
2	2	San Francisco	1500
3	3	New Jersey	1600
4	4	Seattle, Washington	1700
5	5	Toronto	1800
6	6	Sydney	2200
7	7	Mexico City	3200
8	8	Beijing	2000
9	9	Bombay	2100

6.

Detecting (Non existent employee as manager for employee 202)

210 in employees table is non existent but manages employee 202.

```
Select manager_id from hosales.employees
where manager_id not in
(Select employee_id from hosales.employees);
```

Correcting (Remove 210 as manager and make the person manage itself)

```
Create table employees as
Select * from hosales.employees
order by employee_id;

Update employees
Set manager_id = employee_id
where manager_id = 210;
```

Before Cleaning:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
202	Pat	Fay	PFAY	603.123.6666	17/AUG/05	MK_REP	6000	(null)	210	20

After Cleaning:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
202	Pat	Fay	PFAY	603.123.6666	17/AUG/05	MK_REP	6000	(null)	202	20

7.

Detecting (100 has no manager (null))

```
Select *from hosales.employees
where manager_id is NULL;
```

Correcting (Make it manage itself)

```
Update employees
Set manager_id = employee_id
where employee_id = 100;
```

Before Cleaning:

	EMPLOYEE_ID	MANAGER_ID
1	100	(null)
2	101	100
3	102	100
4	103	103
5	104	103
6	105	103
7	106	103
8	107	103
9	108	108
10	109	108
11	110	108
12	111	108

After Cleaning:

	EMPLOYEE_ID	MANAGER_ID
1	100	100
2	101	100
3	102	100
4	103	103
5	104	103
6	105	103
7	106	103
8	107	103
9	108	108
10	109	108
11	110	108
12	111	108

8.

Detecting (employee 106 has negative salary):

```
Select * from hosales.employees
Where salary < 0;
```

Correcting (make salary positive)

```
Update employees
Set salary = abs(salary)
Where employee_id = 106;
```

Before Cleaning:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
106	Valli	Pataballa	-4800

After Cleaning:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
106	Valli	Pataballa	4800

9.

Detecting (employee 178 does not have a department_ID)

Select employee_id from employees
where department_id is NULL;

Correcting (look at other employees that have same job_id)

Employee 178 has job_id SA_REP. All other employees having job_id as SA_REP have department_id of 80. So we can make employee 178's department_ID as 80 as well by looking at other similar employees.

Update employees
Set department_id = 80
where employee_id = 178;

Before Cleaning:

	EMPLOYEE_ID	DEPARTMENT_ID
1	178	(null)

After Cleaning:

EMPLOYEE_ID	DEPARTMENT_ID
178	80

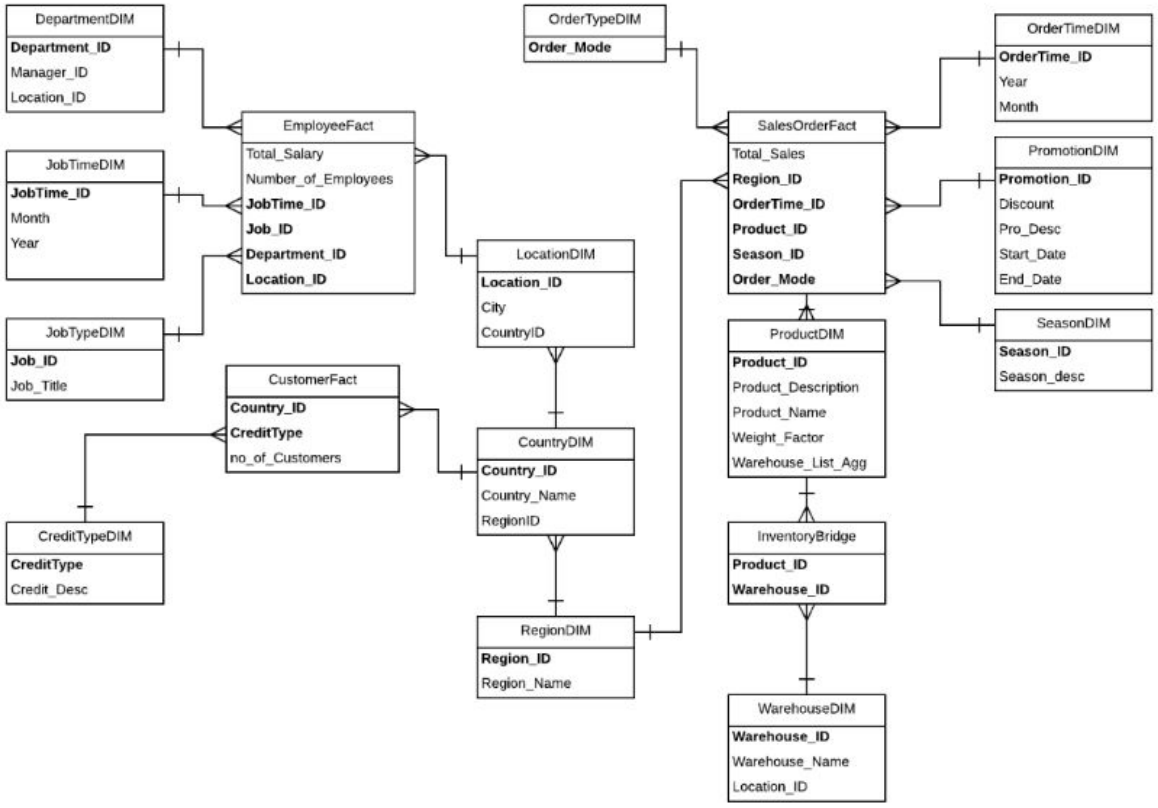
Task 2 Star Schemas

Additional Queries

1. What is the Total Sales in each season (SeasonDIM) for each region (RegionDIM)?
2. Number of low-credit (CreditTypeDIM) customers in each region? (RegionDIM)

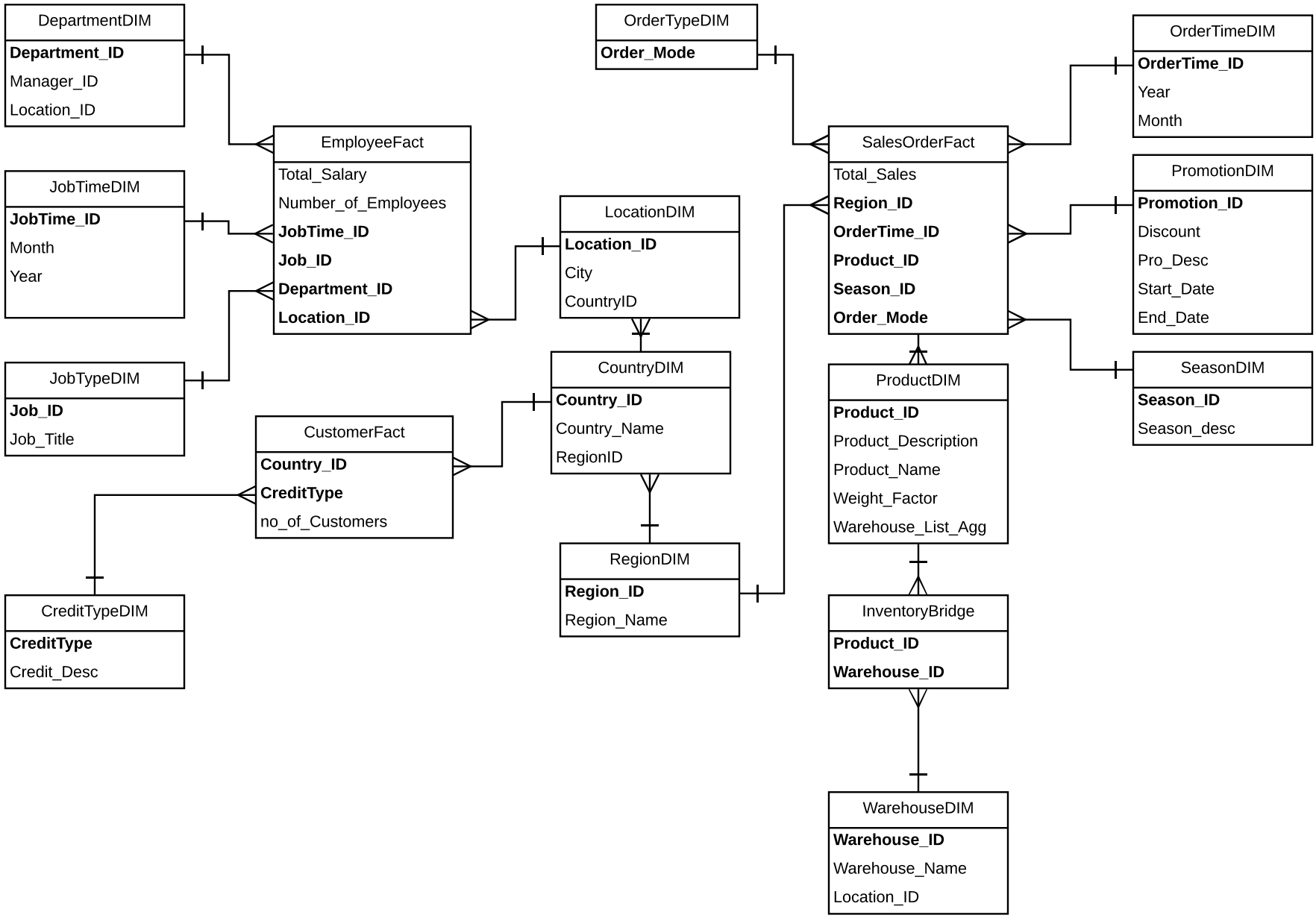
Star Schema

Version-1: Bridge Table and Hierarchy (PDF on next page)

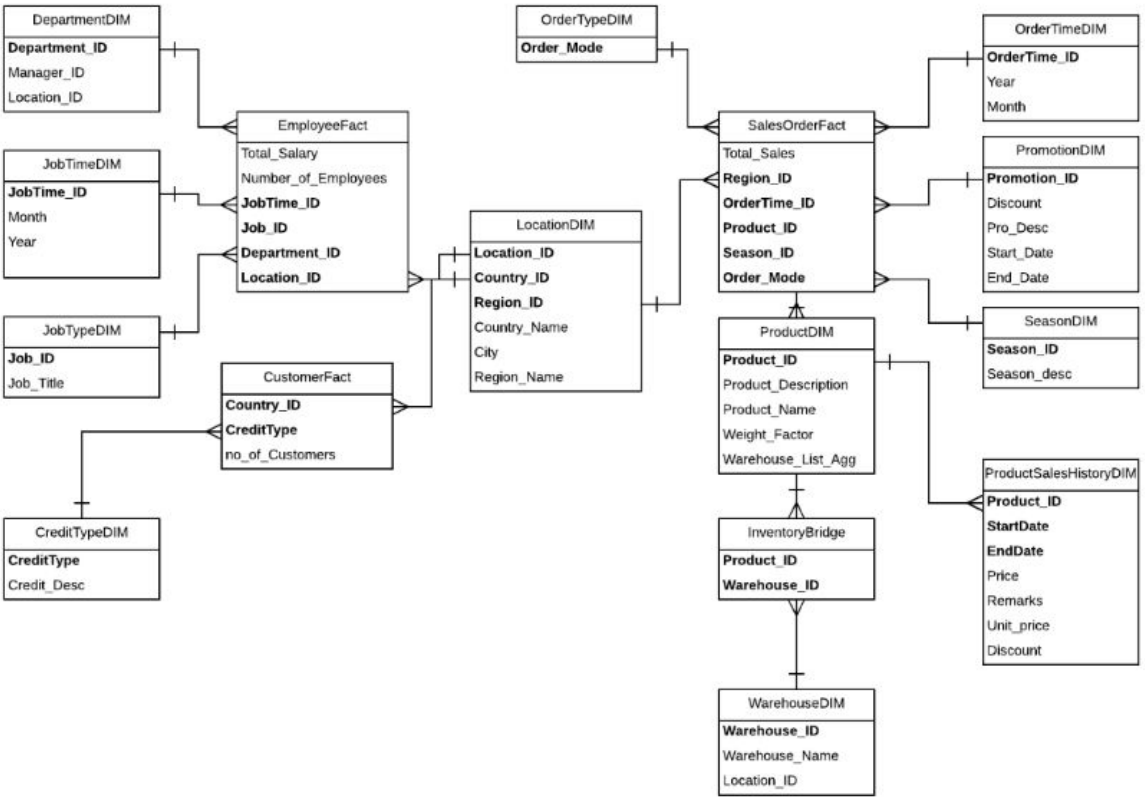


Description:

This Star Schema contains a Bridge table (InventoryBridge) and a Hierarchy (LocationDim, LocationID and RegionDIM). The Bridge Table allows the relation between Warehouse fields and Products via the Warehouse_ID and Product_ID. The Hierarchy splits up all Location based Fields, from specific cities to more general Regions.

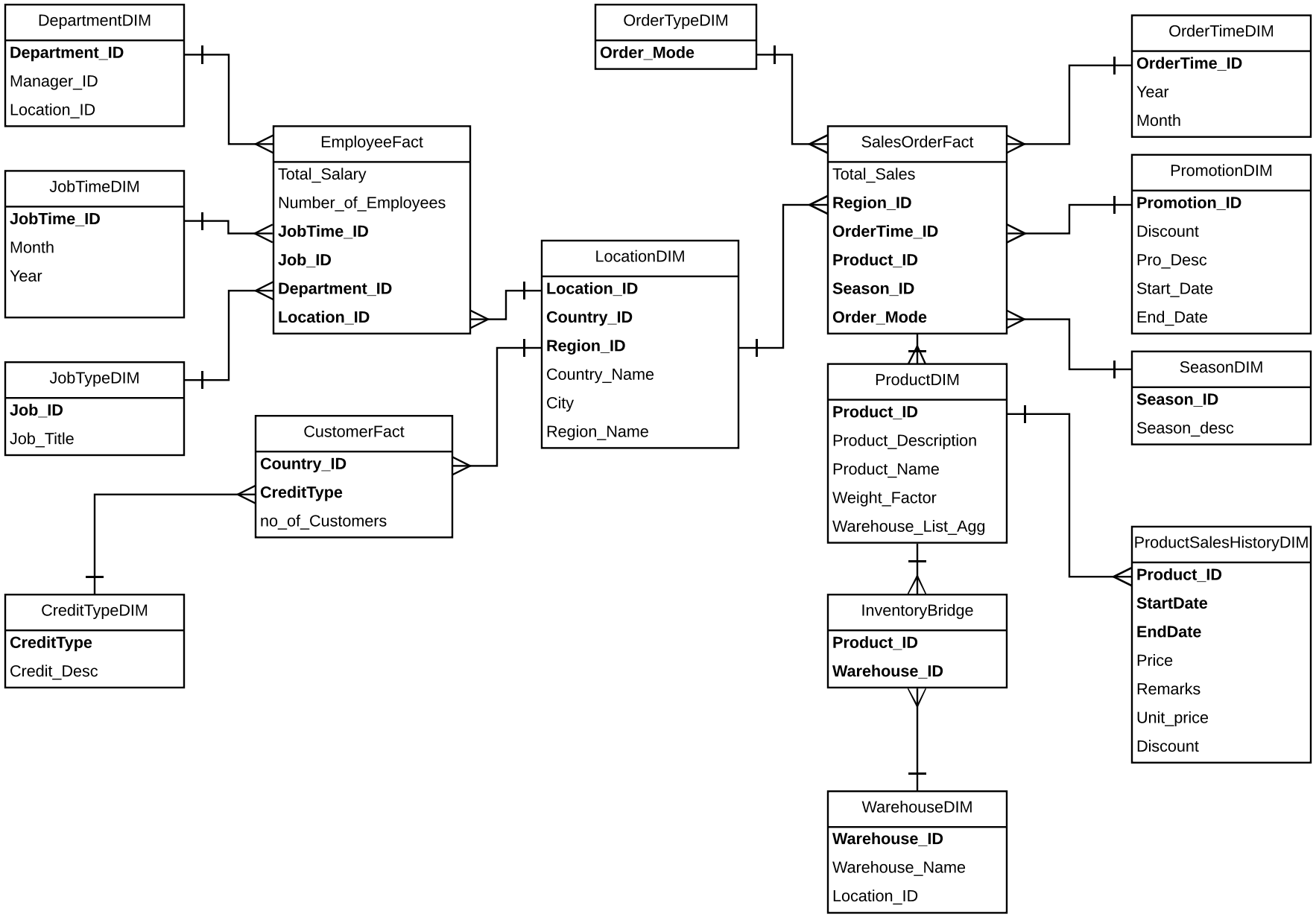

DepartmentDIM
Department_ID
Manager_ID
Location_ID
JobTimeDIM
JobTime_ID
Month
Year
JobTypeDIM
Job_ID
Job_Title
CreditTypeDIM
CreditType
Credit_Desc
EmployeeFact
Total_Salary
Number_of_Employees
JobTime_ID
Job_ID
Department_ID
Location_ID
CustomerFact
Country_ID
CreditType
no_of_Customers
LocationDIM
Location_ID
City
CountryID
CountryDIM
Country_ID
Country_Name
RegionID
RegionDIM
Region_ID
Region_Name
OrderTypeDIM
Order_Mode
SalesOrderFact
Total_Sales
Region_ID
OrderTime_ID
Product_ID
Season_ID
Order_Mode
ProductDIM
Product_ID
Product_Description
Product_Name
Weight_Factor
Warehouse_List_Agg
InventoryBridge
Product_ID
Warehouse_ID
WarehouseDIM
Warehouse_ID
Warehouse_Name
Location_ID
OrderTimeDIM
OrderTime_ID
Year
Month
PromotionDIM
Promotion_ID
Discount
Pro_Desc
Start_Date
End_Date
SeasonDIM
Season_ID
Season_desc

Version-2: Bridge Table and Temporal (PDF on next page)



Description:

This Star-Schema includes a Temporal Dimension ProductSalesHistory and the same Bridge table used in Version-1. The temporal dimension allows management to keep track of the change in sales price of products over time.


DepartmentDIM
Department_ID
Manager_ID
Location_ID
JobTimeDIM
JobTime_ID
Month
Year
JobTypeDIM
Job_ID
Job_Title
CreditTypeDIM
CreditType
Credit_Desc
EmployeeFact
Total_Salary
Number_of_Employees
JobTime_ID
Job_ID
Department_ID
Location_ID
CustomerFact
Country_ID
CreditType
no_of_Customers
LocationDIM
Location_ID
Country_ID
Region_ID
Country_Name
City
Region_Name
OrderTypeDIM
Order_Mode
SalesOrderFact
Total_Sales
Region_ID
OrderTime_ID
Product_ID
Season_ID
Order_Mode
ProductDIM
Product_ID
Product_Description
Product_Name
Weight_Factor
Warehouse_List_Agg
InventoryBridge
Product_ID
Warehouse_ID
WarehouseDIM
Warehouse_ID
Warehouse_Name
Location_ID
OrderTimeDIM
OrderTime_ID
Year
Month
PromotionDIM
Promotion_ID
Discount
Pro_Desc
Start_Date
End_Date
SeasonDIM
Season_ID
Season_desc
ProductSalesHistoryDIM
Product_ID
StartDate
EndDate
Price
Remarks
Unit_price
Discount

Task 3 Star Schema Implementation

Version-1

--Promotion DIM_v1

```
Create Table PromotionDIM_v1 as
Select *
from hosales.promotions;
```

	PROMOTION_ID	DISCOUNT	PRO_DESC	START_DATE	END_DATE
1	1	Full Price	Full Price	(null)	(null)
2	4	30% off	2007 Chrismax Discount	01/DEC/07	31/DEC/07
3	2	20% off	2007 Mid-Season Discount	01/JUN/07	30/JUN/07
4	5	20% off	2008 Mid-Season Discount	01/JUN/08	30/JUN/08
5	3	10% off	2007 VIP Day	10/NOV/07	11/NOV/07

--Order Type DIM_v1

```
Create Table OrderTypeDim_V1 as
select distinct order_mode
from Orders;
```

	ORDER_MODE
1	direct
2	online

--Order Time DIM_v1

```
Create Table OrderTimeDim_V1 as
select distinct
to_char(order_date, 'yyyymm') as orderTime_ID ,
to_char(order_date, 'yyyy') as year,
to_char(order_date, 'mm') as month
From Orders;
```

	ORDERTIME_ID	YEAR	MONTH
1	200709	2007	09
2	200702	2007	02
3	200902	2009	02
4	200611	2006	11
5	200802	2008	02
6	200602	2006	02
7	200607	2006	07
8	200805	2008	05
9	200707	2007	07
10	200703	2007	03
11	200803	2008	03
12	200403	2004	03
13	200704	2007	04
14	200710	2007	10
15	200708	2007	08
16	200601	2006	01
17	200808	2008	08
18	200807	2008	07
19	200801	2008	01
20	200705	2007	05
21	200706	2007	06
22	200806	2008	06
23	200712	2007	12
24	200603	2006	03
25	200609	2006	09
26	200711	2007	11

--ProductDIM_v1

```
create table ProductDim_v1 as
select distinct
p.product_ID, p.product_Description, p.Product_name,
1/count (i.warehouse_ID) as weight_factor,
LISTAGG (i.warehouse_ID,'_') within group (order by i.warehouse_ID) as warehouse_list_agg
from hosales.Products p, hosales.inventories i
where p.product_ID = i.product_ID
```

Group by p.product_ID, p.product_Description, p.Product_name;

	PRODUCT_ID	PRODUCT_DESCRIPT...	PRODUCT_NAME	WEIGHT_F...	WAREHOUSE_LIST_AGG
1	3000	Envoy Laptop, 3...	Laptop 32/10/56	0.111111...	1_2_3_4_5_6_7_8_9
2	3300	Screws: Steel, ...	Screws <S.32.P>	0.1	1_2_2_3_4_5_6_7_8_9
3	3400	8GB capacity SC...	HD 8GB /SE	0.2	2_4_6_8_9
4	1729	Cleaning Chemic...	Chemicals - RCP	0.166666...	3_5_6_7_8_9
5	1733	220V Power supp...	PS 220V /UK	0.111111...	1_2_3_4_5_6_7_8_9
6	1734	10 ft RS232 cab...	Cable RS232 ...	0.111111...	1_2_3_4_5_6_7_8_9
7	1737	10ft SCSI2 F/W ...	Cable SCSI 1...	0.111111...	1_2_3_4_5_6_7_8_9
8	1738	110 V Power Sup...	PS 110V /US	0.111111...	1_2_3_4_5_6_7_8_9
9	1739	SDRAM memory, 1...	SDRAM - 128 MB	0.2	2_4_6_8_9
10	1740	Tape drive - 12...	TD 12GB/DAT	0.2	2_4_6_8_9
11	1742	CD drive, read ...	CD-ROM 500/16x	0.2	2_4_6_8_9
12	1745	20ft SCSI2 Wide...	Cable SCSI 2...	0.111111...	1_2_3_4_5_6_7_8_9
13	1748	220 Volt Power ...	PS 220V /EUR	0.111111...	1_2_3_4_5_6_7_8_9
14	1749	Memory DIMM: RA...	DIMM - 256MB	0.2	2_4_6_8_9
15	1750	Memory DIMM: RA...	DIMM - 2GB	0.2	2_4_6_8_9
16	1755	32MB Non-Mirror...	32MB Cache /NM	0.2	2_4_6_8_9
17	1763	Memory DIMM: RA...	DIMM - 64MB	0.2	2_4_6_8_9
18	1768	Hard drive disk...	HD 8.2GB @5400	0.2	2_4_6_8_9
19	1769	Graphics proces...	GP 800x600	0.2	2_4_6_8_9
20	1770	8MB Non-Mirrore...	8MB Cache /NM	0.2	2_4_6_8_9
21	1772	Hard disk drive...	HD 9.1GB @10000	0.2	2_4_6_8_9
22	1774	Base ISO Commun...	Base ISO CP ...	0.2	2_5_6_7_8
23	1775	ISO Communicati...	Client ISO C...	0.2	2_5_6_7_8
24	1778	C programming s...	C for SPNIX3...	0.2	2_5_6_7_8
25	1779	C programming l...	C for SPNIX3...	0.2	2_5_6_7_8
26	1780	C programming s...	C for SPNIX3...	0.2	2_5_6_7_8
27	1781	CD Writer, read...	CDW 20/48/E	0.2	2_4_6_8_9
28	1787	Dual CPU @ 300M...	CPU D300	0.2	2_4_6_8_9
29	1788	Dual CPU @ 600M...	CPU D600	0.2	2_4_6_8_9
30	1791	700 characters ...	Industrial 7...	0.333333...	6_8_9

--Credit Type DIM_v1

Create Table_v1 CreditTypeDIM_v1
(creditType varchar2(20),
credit_desc varchar2(30));

Insert into creditTypeDIM_v1 values
('Low', 'credit <= 1500');
Insert into creditTypeDIM_v1 values
('Med', '1500 < credit <= 3500');
Insert into creditTypeDIM_v1 values
('High', 'credit > 3500');

	CREDITTYPE	CREDIT_DESC
1	Low	credit <= 1500
2	Med	1500 < credit <= 3500
3	High	credit > 3500

Using constraints (OPTIONAL WAY):

```
Create Table CreditTypeDIM_v1
As Select * From
(
Select
(Case
   when Credit_Limit <= 1500 Then 'Low'
   When Credit_Limit BETWEEN 1500 and 3500 Then 'Medium'
   Else 'High'
 END) As CreditType,
 (Case
   when Credit_Limit <= 1500 Then 'Credit <= 1500'
   When Credit_Limit BETWEEN 1500 and 3500 Then '1500 < Credit <= 3500'
   Else 'Credit > 3500'
 END) As CreditDesc
From Customers
)
Group by CreditType, CreditDesc
Order by CreditType;
```

	CREDITTYPE	CREDITDESC
1	High	Credit > 3500
2	Low	Credit <= 1500
3	Medium	1500 < Credit <= 3500

--CountryDIM_v1

```
Create table CountryDIM_v1
As select * from hosales.countries;
```

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
1	AR	Argentina	2
2	AU	Australia	3
3	BE	Belgium	1
4	BR	Brazil	2
5	CA	Canada	2
6	CH	Switzerland	1
7	CN	China	3
8	DE	Germany	1
9	DK	Denmark	1
10	EG	Egypt	4
11	FR	France	1
12	IL	Israel	4
13	IN	India	3
14	IT	Italy	1
15	JP	Japan	3
16	KW	Kuwait	4
17	ML	Malaysia	3
18	MX	Mexico	2
19	NG	Nigeria	4
20	NL	Netherlands	1
21	SG	Singapore	3
22	TH	Thailand	3
23	UK	United Kingdom	1
24	US	United States of America	2
25	ZM	Zambia	4
26	ZW	Zimbabwe	4

--RegionDIM_v1

Create table regionDIM_v1
As select * from hosales.regions;

	REGION_ID	REGION_NAME
1	1	Europe
2	2	Americas
3	3	Asia
4	4	Middle East and Africa

--LocationDIM_v1

Create table locationDIM_v1
As select distinct
location_ID, city, country_ID
From hosales.locations;

	LOCATION_ID	CITY	COUNTRY_ID
1	1000	Roma	IT
2	1100	Venice	IT
3	1200	Tokyo	JP
4	1300	Hiroshima	JP
5	1400	Southlake	US
6	1500	South San Francisco	US
7	1600	South Brunswick	US
8	1700	Seattle	US
9	1800	Toronto	CA
10	1900	Whitehorse	CA
11	2000	Beijing	CN
12	2100	Bombay	IN
13	2200	Sydney	AU
14	2300	Singapore	SG
15	2400	London	UK
16	2500	Oxford	UK
17	2600	Stretford	UK
18	2700	Munich	DE
19	2800	Sao Paulo	BR
20	2900	Geneva	CH
21	3000	Bern	CH
22	3100	Utrecht	NL
23	3200	Mexico City	MX

--JobTypeDIM_v1

```
Create table jobTypeDIM_v1
As select distinct
Job_id, job_title
From hosales.jobs;
```

	JOB_ID	JOB_TITLE
1	AD_PRES	President
2	AD_VP	Administration Vice President
3	AD_ASST	Administration Assistant
4	FI_MGR	Finance Manager
5	FI_ACCOUNT	Accountant
6	AC_MGR	Accounting Manager
7	AC_ACCOUNT	Public Accountant
8	SA_MAN	Sales Manager
9	SA_REP	Sales Representative
10	PU_MAN	Purchasing Manager
11	PU_CLERK	Purchasing Clerk
12	ST_MAN	Stock Manager
13	ST_CLERK	Stock Clerk
14	SH_CLERK	Shipping Clerk
15	IT_PROG	Programmer
16	MK_MAN	Marketing Manager
17	MK_REP	Marketing Representative
18	HR_REP	Human Resources Representative
19	PR_REP	Public Relations Representative

--DepartmentDIM_v1

Create table departmentDIM_v1

As select * from hosales.departments;

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
9	90	Executive	100	1700
10	100	Finance	108	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700
14	140	Control And Credit	(null)	1700
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700

--JobTimeDIM_v1

```
Create table jobTimeDIM_v1
As select distinct
to_char(hire_date, 'yyyymm') as jobTimeID ,
to_char(hire_date, 'yyyy') as year,
to_char(hire_date, 'mm') as month
From employees;
```

	JOBTIMEID	YEAR	MONT
1	200702	2007	02
2	200512	2005	12
3	200510	2005	10
4	200307	2003	07
5	200604	2006	04
6	200501	2005	01
7	200306	2003	06
8	200611	2006	11
9	200804	2008	04
10	200602	2006	02
11	200802	2008	02
12	200607	2006	07
13	200408	2004	08
14	200101	2001	01
15	200504	2005	04
16	200803	2008	03
17	200704	2007	04
18	200310	2003	10
19	200503	2005	03
20	200401	2004	01
21	200403	2004	03
22	200703	2007	03
23	200405	2004	05
24	200506	2005	06
25	200305	2003	05
26	200507	2005	07
27	200701	2007	01
28	200608	2006	08
29	200612	2006	12

--InventoryBridge_v1

Create table InventoryBridge_v1
As select * from hosales.inventories;

	PRODUCT_ID	WAREHOUSE_ID
1	1733	1
2	1734	1
3	1737	1
4	1738	1
5	1745	1
6	1748	1
7	2278	1
8	2316	1
9	2319	1
10	2322	1
11	2323	1
12	2370	1
13	2373	1
14	2377	1
15	2380	1
16	2387	1
17	2408	1
18	2418	1
19	2419	1
20	2457	1
21	2638	1
22	2878	1
23	2879	1
24	3000	1
25	3003	1
26	3004	1
27	3099	1
28	3124	1

--WarehouseDIM_v1

Create table WarehouseDIM_v1
As select warehouse_id, warehouse_name, location_id
From warehouses;

	WAREHOUSE_ID	WAREHOUSE_NAME	LOCATION_ID
1	1	Southlake, Texas	1400
2	2	San Francisco	1500
3	3	New Jersey	1600
4	4	Seattle, Washington	1700
5	5	Toronto	1800
6	6	Sydney	2200
7	7	Mexico City	3200
8	8	Beijing	2000
9	9	Bombay	2100

--SeasonDIM_v1

```
Create table seasonDIM_v1
(season_ID number,
seasondesc varchar2(20));

Insert into seasonDIM_v1 values
(1, 'Summer');
Insert into seasonDIM_v1 values
(2, 'Winter');
Insert into seasonDIM_v1 values
(3, 'Spring');
Insert into seasonDIM_v1 values
(4, 'Autumn');
```

	SEASON_ID	SEASONDESC
1	1	Summer
2	2	Winter
3	3	Spring
4	4	Autumn

--CustomerfactTemp_v1 + FACT

```
Create table customerfactTemp_v1 as select
C.country_id, cu.credit_limit, cu.customer_id
From hosales.countries c, customers cu
Where c.country_id = cu.country_id;

Alter table customerfactTemp_v1
Add (creditType varchar(20));

Update customerfactTemp_v1
Set creditType = 'Low'
Where Credit_Limit <= 1500;

Update customerfactTemp_v1
Set creditType = 'Med'
Where Credit_Limit > 1500 AND Credit_limit <= 3500;

Update customerfactTemp_v1
Set creditType = 'High'
Where Credit_Limit > 3500;


Create table customerfact_v1 as select
Country_id, creditType, count(customer_id) as no_of_customers
From customerfactTemp_v1
```

Group by Country_id, creditType;

	COUNTRY_ID	CREDITTYPE	NO_OF_CUSTOMERS
1	JP	Med	1
2	TH	High	1
3	CN	Low	3
4	IT	Low	29
5	IT	Med	6
6	IN	High	23
7	CH	Low	12
8	CA	Low	1
9	IN	Low	23
10	US	Low	115
11	IT	High	8
12	TH	Low	1
13	US	High	27
14	IN	Med	8
15	CH	Med	14
16	CH	High	3
17	US	Med	42
18	DE	Med	3
19	CN	High	1

--EmployeefactTemp_v1 + FACT

```
Create table EmployeeFact_v1 as select
to_char(e.hire_date, 'yyyymm') as jobTimeID, j.job_id, d.department_id,
l.location_ID, sum(e.salary) as total_salary, count(e.employee_ID) as number_of_employees
FROM hosales.jobs j, employees e, hosales.departments d, hosales.locations l
WHERE j.job_id = e.job_id and e.department_ID = d.department_ID and d.location_ID =
l.location_ID
GROUP BY to_char(hire_date, 'yyyymm'), j.job_id, d.department_id, l.location_ID;
```

	JOBTIMEID	JOB_ID	DEPARTMENT_ID	LOCATION_ID	TOTAL_SALARY	NUMBER_OF_EMPLOYEES
1	200402	MK_MAN	20	1800	13000	1
2	200212	PU_MAN	30	1700	11000	1
3	200310	ST_CLERK	50	1500	3500	1
4	200603	ST_CLERK	50	1500	2600	1
5	200607	SH_CLERK	50	1500	6100	2
6	200503	SH_CLERK	50	1500	3900	1
7	200604	SH_CLERK	50	1500	3100	1
8	200605	SH_CLERK	50	1500	3000	1
9	200503	SA_MAN	80	2500	12000	1
10	200501	SA_REP	80	2500	10000	1
11	200503	SA_REP	80	2500	37800	4
12	200508	SA_REP	80	2500	9000	1
13	200603	SA_REP	80	2500	26600	3
14	200711	SA_REP	80	2500	7000	1
15	200512	SA_REP	80	2500	7500	1
16	200509	AD_VP	90	1700	17000	1
17	200101	AD_VP	90	1700	17000	1
18	200508	MK_REP	20	1800	6000	1
19	200305	PU_CLERK	30	1700	3100	1
20	200305	ST_MAN	50	1500	7900	1
21	200604	ST_CLERK	50	1500	2500	1
22	200901	SH_CLERK	50	1500	3200	1
23	200502	SH_CLERK	50	1500	4100	1
24	200501	SA_MAN	80	2500	13500	1
25	200604	SA_REP	80	2500	8400	1
26	200701	ST_CLERK	50	1500	2400	1
27	200607	ST_CLERK	50	1500	2500	1
28	200602	SH_CLERK	50	1500	3100	1

--SalesOrderfactTemp_v1 + FACT

```
Create table salesOrderfactTemp_v1 as select
o.total_price, o.order_mode, oi.product_id, to_char(o.order_date, 'yyyymm') as orderTime_ID,
c.region_ID, o.order_date as transaction_date, p.start_date, p.end_date
From orders o, hosales.order_items oi, hosales.countries c, customers ct, hosales.promotions p
Where oi.order_id = o.order_id AND  o.customer_ID = ct.customer_ID AND ct.country_ID =
c.country_ID AND o.promotion_id = p.promotion_ID;

alter table salesOrderfactTemp_v1
add (season_Id number);

update salesOrderfactTemp_v1
set season_Id = 1
where to_char (transaction_date, 'mm')
in ('12','01','02');
```

```
update salesOrderfactTemp_v1
set season_Id = 2
where to_char (transaction_date, 'mm')
in ('03','04','05');

update salesOrderfactTemp_v1
set season_Id = 3
where to_char (transaction_date, 'mm')
in ('06','07','08');

update salesOrderfactTemp_v1
set season_Id = 4
where to_char (transaction_date, 'mm')
in ('09','10','11');

Create Table salesOrderfact_v1 as select
Order_mode, product_id, orderTime_ID, region_ID, season_ID,
Sum (total_price) as total_sales
From salesOrderfactTemp_v1
Where transaction_date >= start_date AND transaction_date <= end_date
Group by Order_mode, product_id, orderTime_ID, region_ID, season_ID;
```

	ORDER_MODE	PRODUCT_ID	ORDERTIME_ID	REGION_ID	SEASON_ID	TOTAL_SALES
1	online	2289	200806	2	3	115444.48
2	online	2289	200712	2	1	209503.77
3	direct	1910	200712	2	1	7338.52
4	online	2293	200706	2	3	236221.92
5	online	3165	200706	2	3	8891.52
6	direct	3127	200711	2	4	58381.38
7	direct	3133	200711	2	4	58381.38
8	direct	1948	200712	2	1	7338.52
9	direct	3150	200712	2	1	7824.67
10	online	2335	200712	2	1	209503.77
11	online	2350	200712	2	1	209503.77
12	online	1910	200806	2	3	100.8
13	direct	2810	200706	2	3	38.4
14	direct	3106	200711	2	4	58381.38
15	online	2276	200706	2	3	9106.48
16	online	3117	200806	2	3	47176
17	direct	3114	200711	2	4	12869.64
18	online	2311	200706	2	3	39951.12
19	online	2302	200712	2	1	209503.77
20	direct	3127	200712	2	1	7824.67
21	online	2311	200712	2	1	209503.77
22	direct	3139	200706	2	3	1140.8
23	direct	3143	200706	2	3	1140.8
24	online	3187	200706	2	3	8891.52
25	online	2326	200706	2	3	39951.12
26	online	2330	200706	2	3	257960.08
27	online	3193	200706	2	3	8891.52
28	direct	3170	200711	2	4	12869.64

Version-2

--Promotion DIM_v2

Create Table PromotionDIM_v2 as
Select *
from hosales.promotions;

	PROMOTION_ID	DISCOUNT	PRO_DESC	START_DATE	END_DATE
1	1	Full Price	Full Price	(null)	(null)
2	4	30% off	2007 Chrismax Discount	01/DEC/07	31/DEC/07
3	2	20% off	2007 Mid-Season Discount	01/JUN/07	30/JUN/07
4	5	20% off	2008 Mid-Season Discount	01/JUN/08	30/JUN/08
5	3	10% off	2007 VIP Day	10/NOV/07	11/NOV/07

--Order Type DIM_v2

Create Table OrderTypeDim_v2 as
select distinct order_mode
from Orders;

	ORDER_MODE
1	direct
2	online

--Order Time DIM_v2

Create Table OrderTimeDim_v2 as
select distinct
to_char(order_date, 'yyyymm') as orderTime_ID ,
to_char(order_date, 'yyyy') as year,
to_char(order_date, 'mm') as month
From orders;

	ORDERTIME_ID	YEAR	MONTH
1	200709	2007	09
2	200702	2007	02
3	200902	2009	02
4	200611	2006	11
5	200802	2008	02
6	200602	2006	02
7	200607	2006	07
8	200805	2008	05
9	200707	2007	07
10	200703	2007	03
11	200803	2008	03
12	200403	2004	03
13	200704	2007	04
14	200710	2007	10
15	200708	2007	08
16	200601	2006	01
17	200808	2008	08
18	200807	2008	07
19	200801	2008	01
20	200705	2007	05
21	200706	2007	06
22	200806	2008	06
23	200712	2007	12
24	200603	2006	03
25	200609	2006	09
26	200711	2007	11

--ProductDIM_v2

create table ProductDim_v2 as

```
select distinct
p.product_ID, p.product_Description, p.Product_name,
1/count (i.warehouse_ID) as weight_factor,
LISTAGG (i.warehouse_ID,'_') within group (order by i.warehouse_ID) as warehouse_list_agg
from hosales.Products p, hosales.inventories i
where p.product_ID = i.product_ID
Group by p.product_ID, p.product_Description, p.Product_name;
```

	PRODUCT_ID	PRODUCT_DESCRIPT...	PRODUCT_NAME	WEIGHT_F...	WAREHOUSE_LIST_AGG
1	3000	Envoy Laptop, 3...	Laptop 32/10/56	0.111111...	1_2_3_4_5_6_7_8_9
2	3300	Screws: Steel, ...	Screws <S.32.P>	0.1	1_2_2_3_4_5_6_7_8_9
3	3400	8GB capacity SC...	HD 8GB /SE	0.2	2_4_6_8_9
4	1729	Cleaning Chemic...	Chemicals - RCP	0.166666...	3_5_6_7_8_9
5	1733	220V Power supp...	PS 220V /UK	0.111111...	1_2_3_4_5_6_7_8_9
6	1734	10 ft RS232 cab...	Cable RS232 ...	0.111111...	1_2_3_4_5_6_7_8_9
7	1737	10ft SCSI2 F/W ...	Cable SCSI 1...	0.111111...	1_2_3_4_5_6_7_8_9
8	1738	110 V Power Sup...	PS 110V /US	0.111111...	1_2_3_4_5_6_7_8_9
9	1739	SDRAM memory, 1...	SDRAM - 128 MB	0.2	2_4_6_8_9
10	1740	Tape drive - 12...	TD 12GB/DAT	0.2	2_4_6_8_9
11	1742	CD drive, read ...	CD-ROM 500/16x	0.2	2_4_6_8_9
12	1745	20ft SCSI2 Wide...	Cable SCSI 2...	0.111111...	1_2_3_4_5_6_7_8_9
13	1748	220 Volt Power ...	PS 220V /EUR	0.111111...	1_2_3_4_5_6_7_8_9
14	1749	Memory DIMM: RA...	DIMM - 256MB	0.2	2_4_6_8_9
15	1750	Memory DIMM: RA...	DIMM - 2GB	0.2	2_4_6_8_9
16	1755	32MB Non-Mirror...	32MB Cache /NM	0.2	2_4_6_8_9
17	1763	Memory DIMM: RA...	DIMM - 64MB	0.2	2_4_6_8_9
18	1768	Hard drive disk...	HD 8.2GB @5400	0.2	2_4_6_8_9
19	1769	Graphics proces...	GP 800x600	0.2	2_4_6_8_9
20	1770	8MB Non-Mirrore...	8MB Cache /NM	0.2	2_4_6_8_9
21	1772	Hard disk drive...	HD 9.1GB @10000	0.2	2_4_6_8_9
22	1774	Base ISO Commun...	Base ISO CP ...	0.2	2_5_6_7_8
23	1775	ISO Communicati...	Client ISO C...	0.2	2_5_6_7_8
24	1778	C programming s...	C for SPNIX3...	0.2	2_5_6_7_8
25	1779	C programming l...	C for SPNIX3...	0.2	2_5_6_7_8
26	1780	C programming s...	C for SPNIX3...	0.2	2_5_6_7_8
27	1781	CD Writer, read...	CDW 20/48/E	0.2	2_4_6_8_9
28	1787	Dual CPU @ 300M...	CPU D300	0.2	2_4_6_8_9
29	1788	Dual CPU @ 600M...	CPU D600	0.2	2_4_6_8_9
30	1791	700 characters ...	Industrial 7...	0.333333...	6_8_9

--Credit Type DIM_v2

```
Create Table_v2 CreditTypeDIM_v2
(creditType varchar2(20),
credit_desc varchar2(30));

Insert into creditTypeDIM_v2 values
('Low', 'credit <= 1500');
```

Insert into creditTypeDIM_v2 values
('Med', '1500 < credit <= 3500');
Insert into creditTypeDIM_v2 values
('High', 'credit > 3500');

	CREDITTYPE	CREDIT_DESC
1	Low	credit <= 1500
2	Med	1500 < credit <= 3500
3	High	credit > 3500

--LocationDIm_v2

```
create table LocationDIM_v2 as
Select l.location_ID, l.city, c.country_id,
    c.country_name, r.region_id, r.region_name
from hosales.Locations l, hosales.Countries c, hosales.regions r
where r.region_id = c.region_id and c.country_id = l.country_id;
```

	LOCATION_ID	CITY	COUNTRY_ID	COUNTRY_NAME	REGION_ID	REGION_NAME
1	3100	Utrecht	NL	Netherlands	1	Europe
2	3000	Bern	CH	Switzerland	1	Europe
3	2900	Geneva	CH	Switzerland	1	Europe
4	2700	Munich	DE	Germany	1	Europe
5	2600	Stretford	UK	United Kingdom	1	Europe
6	2500	Oxford	UK	United Kingdom	1	Europe
7	2400	London	UK	United Kingdom	1	Europe
8	1100	Venice	IT	Italy	1	Europe
9	1000	Roma	IT	Italy	1	Europe
10	3200	Mexico City	MX	Mexico	2	Americas
11	2800	Sao Paulo	BR	Brazil	2	Americas
12	1900	Whitehorse	CA	Canada	2	Americas
13	1800	Toronto	CA	Canada	2	Americas
14	1700	Seattle	US	United States of America	2	Americas
15	1600	South Brunswick	US	United States of America	2	Americas
16	1500	South San Francisco	US	United States of America	2	Americas
17	1400	Southlake	US	United States of America	2	Americas
18	2300	Singapore	SG	Singapore	3	Asia
19	2200	Sydney	AU	Australia	3	Asia
20	2100	Bombay	IN	India	3	Asia
21	2000	Beijing	CN	China	3	Asia
22	1300	Hiroshima	JP	Japan	3	Asia
23	1200	Tokyo	JP	Japan	3	Asia

--JobTypeDIM_v2

```
Create table jobTypeDIM_v2
As select distinct
```

Job_id, job_title
From hosales.jobs;

	JOB_ID	JOB_TITLE
1	AD_PRES	President
2	AD_VP	Administration Vice President
3	AD_ASST	Administration Assistant
4	FI_MGR	Finance Manager
5	FI_ACCOUNT	Accountant
6	AC_MGR	Accounting Manager
7	AC_ACCOUNT	Public Accountant
8	SA_MAN	Sales Manager
9	SA_REP	Sales Representative
10	PU_MAN	Purchasing Manager
11	PU_CLERK	Purchasing Clerk
12	ST_MAN	Stock Manager
13	ST_CLERK	Stock Clerk
14	SH_CLERK	Shipping Clerk
15	IT_PROG	Programmer
16	MK_MAN	Marketing Manager
17	MK_REP	Marketing Representative
18	HR_REP	Human Resources Representative
19	PR_REP	Public Relations Representative

--DepartmentDIM_v2
Create table departmentDIM_v2
As select * from hosales.departments;

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
9	90	Executive	100	1700
10	100	Finance	108	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700
14	140	Control And Credit	(null)	1700
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700

--JobTimeDIM_v2

```
Create table jobTimeDIM_v2
As select distinct
to_char(hire_date, 'yyyymm') as jobTimeID ,
to_char(hire_date, 'yyyy') as year,
to_char(hire_date, 'mm') as month
From employees;
```

	JOBTIMEID	YEAR	MONT
1	200702	2007	02
2	200512	2005	12
3	200510	2005	10
4	200307	2003	07
5	200604	2006	04
6	200501	2005	01
7	200306	2003	06
8	200611	2006	11
9	200804	2008	04
10	200602	2006	02
11	200802	2008	02
12	200607	2006	07
13	200408	2004	08
14	200101	2001	01
15	200504	2005	04
16	200803	2008	03
17	200704	2007	04
18	200310	2003	10
19	200503	2005	03
20	200401	2004	01
21	200403	2004	03
22	200703	2007	03
23	200405	2004	05
24	200506	2005	06
25	200305	2003	05
26	200507	2005	07
27	200701	2007	01
28	200608	2006	08
29	200612	2006	12

--ProductSalesHistoryDIM_v2

```
create table ProductSalesHistoryDIM_v2
as select distinct
o.product_id, p.start_date, p.end_date, p.pro_desc as remarks,o.unit_price, p.DISCOUNT
from hosales.order_items o, orders k, hosales.promotions p
where o.order_id = k.order_id and k.promotion_id = p.promotion_id;

alter table ProductSalesHistoryDIM_v2
Add (Price number);

update ProductSalesHistoryDIM_v2
Set price = (0.8 * Unit_price)
Where discount = ‘20% off’;

update ProductSalesHistoryDIM_v2
```

```
Set price = (0.7 * Unit_price)
Where discount = '30% off';

update ProductSalesHistoryDIM_v2
Set price = (0.8 * Unit_price)
Where discount = '20% off';

update ProductSalesHistoryDIM_v2
Set price = Unit_price
Where discount = 'Full Price';

update ProductSalesHistoryDIM_v2
Set price = (0.9 * Unit_price)
Where discount = '10% off';
```

	PRODUCT_ID	START_DATE	END_DATE	REMARKS	UNIT_PRICE	DISCOUNT	PRICE
1	2211	(null)	(null)	Full Price	3.3	Full Price	3.3
2	1787	(null)	(null)	Full Price	101	Full Price	101
3	2522	01/JUN/07	30/JUN/07	2007 Mid-Season Discount	40	20% off	32
4	1782	(null)	(null)	Full Price	125	Full Price	125
5	2761	01/JUN/07	30/JUN/07	2007 Mid-Season Discount	26	20% off	20.8
6	3133	(null)	(null)	Full Price	43	Full Price	43
7	3234	(null)	(null)	Full Price	32	Full Price	32
8	2308	01/JUN/08	30/JUN/08	2008 Mid-Season Discount	56	20% off	44.8
9	2093	(null)	(null)	Full Price	7.7	Full Price	7.7
10	2319	01/JUN/07	30/JUN/07	2007 Mid-Season Discount	25	20% off	20
11	3167	(null)	(null)	Full Price	54	Full Price	54
12	2268	(null)	(null)	Full Price	75	Full Price	75
13	2350	01/DEC/07	31/DEC/07	2007 Chrismax Discount	2341.9	30% off	1639.33
14	2350	01/JUN/07	30/JUN/07	2007 Mid-Season Discount	2341.9	20% off	1873.52
15	2808	(null)	(null)	Full Price	2	Full Price	2
16	3252	(null)	(null)	Full Price	25	Full Price	25
17	3173	10/NOV/07	11/NOV/07	2007 VIP Day	80	10% off	72
18	2264	(null)	(null)	Full Price	199.1	Full Price	199.1
19	3106	(null)	(null)	Full Price	48	Full Price	48
20	3150	01/JUN/07	30/JUN/07	2007 Mid-Season Discount	17	20% off	13.6
21	1820	(null)	(null)	Full Price	52	Full Price	52
22	2976	(null)	(null)	Full Price	52	Full Price	52
23	2471	(null)	(null)	Full Price	482.9	Full Price	482.9
24	2492	(null)	(null)	Full Price	43	Full Price	43
25	2638	(null)	(null)	Full Price	137	Full Price	137
26	2810	01/JUN/07	30/JUN/07	2007 Mid-Season Discount	6	20% off	4.8
27	3082	(null)	(null)	Full Price	81	Full Price	81

--InventoryBridge_v2

```
Create table InventoryBridge_v2
As select * from hosales.inventories;
```

	PRODUCT_ID	WAREHOUSE_ID
1	1733	1
2	1734	1
3	1737	1
4	1738	1
5	1745	1
6	1748	1
7	2278	1
8	2316	1
9	2319	1
10	2322	1
11	2323	1
12	2370	1
13	2373	1
14	2377	1
15	2380	1
16	2387	1
17	2408	1
18	2418	1
19	2419	1
20	2457	1
21	2638	1
22	2878	1
23	2879	1
24	3000	1
25	3003	1
26	3004	1
27	3099	1
28	3124	1

--WarehouseDIM_v2

Create table WarehouseDIM_v2
As select * from warehouses;

	WAREHOUSE_ID	WAREHOUSE_NAME	LOCATION_ID
1	1	Southlake, Texas	1400
2	2	San Francisco	1500
3	3	New Jersey	1600
4	4	Seattle, Washington	1700
5	5	Toronto	1800
6	6	Sydney	2200
7	7	Mexico City	3200
8	8	Beijing	2000
9	9	Bombay	2100

--SeasonDIM_v2

```
Create table seasonDIM_v2
(season_ID number,
seasondesc varchar2(20));

Insert into seasonDIM_v2 values
(1, 'Summer');
Insert into seasonDIM_v2 values
(2, 'Winter');
Insert into seasonDIM_v2 values
(1, 'Spring');
Insert into seasonDIM_v2 values
(1, 'Autumn');
```

	SEASON_ID	SEASONDESC
1	1	Summer
2	2	Winter
3	3	Spring
4	4	Autumn

--CustomerfactTemp_v2 + FACT

```
Create table customerfactTemp_v2 as select
C.country_id, cu.credit_limit, cu.customer_id
From hosales.countries c, customers cu
Where c.country_id = cu.country_id;

Alter table customerfactTemp_v2
Add (creditType varchar(20));

Update customerfactTemp_v2
Set creditType = 'Low'
Where Credit_Limit <= 1500;

Update customerfactTemp_v2
Set creditType = 'Med'
Where Credit_Limit > 1500 AND Credit_limit <= 3500;

Update customerfactTemp_v2
Set creditType = 'High'
Where Credit_Limit > 3500;


Create table customerfact_v2 as select
Country_id, creditType, count(customer_id) as no_of_customers
From customerfactTemp_v2
Group by Country_id, creditType;
```

	COUNTRY_ID	CREDITTYPE	NO_OF_CUSTOMERS
1	JP	Med	1
2	TH	High	1
3	CN	Low	3
4	IT	Low	29
5	IT	Med	6
6	IN	High	23
7	CH	Low	12
8	CA	Low	1
9	IN	Low	23
10	US	Low	115
11	IT	High	8
12	TH	Low	1
13	US	High	27
14	IN	Med	8
15	CH	Med	14
16	CH	High	3
17	US	Med	42
18	DE	Med	3
19	CN	High	1

--EmployeefactTemp_v2 + FACT

```
Create table EmployeeFact_v2 as select
to_char(e.hire_date, 'yyyymm') as jobTimeID, j.job_id, d.department_id,
l.location_ID, sum(e.salary) as total_salary, count(e.employee_ID) as number_of_employees
FROM hosales.jobs j, employees e, hosales.departments d, hosales.locations l
WHERE j.job_id = e.job_id and e.department_ID = d.department_ID and d.location_ID =
l.location_ID
GROUP BY to_char(hire_date, 'yyyymm'), j.job_id, d.department_id, l.location_ID;
```

	JOBTIMEID	JOB_ID	DEPARTMENT_ID	LOCATION_ID	TOTAL_SALARY	NUMBER_OF_EMPLOYEES
1	200402	MK_MAN	20	1800	13000	1
2	200212	PU_MAN	30	1700	11000	1
3	200310	ST_CLERK	50	1500	3500	1
4	200603	ST_CLERK	50	1500	2600	1
5	200607	SH_CLERK	50	1500	6100	2
6	200503	SH_CLERK	50	1500	3900	1
7	200604	SH_CLERK	50	1500	3100	1
8	200605	SH_CLERK	50	1500	3000	1
9	200503	SA_MAN	80	2500	12000	1
10	200501	SA_REP	80	2500	10000	1
11	200503	SA_REP	80	2500	37800	4
12	200508	SA_REP	80	2500	9000	1
13	200603	SA_REP	80	2500	26600	3
14	200711	SA_REP	80	2500	7000	1
15	200512	SA_REP	80	2500	7500	1
16	200509	AD_VP	90	1700	17000	1
17	200101	AD_VP	90	1700	17000	1
18	200508	MK_REP	20	1800	6000	1
19	200305	PU_CLERK	30	1700	3100	1
20	200305	ST_MAN	50	1500	7900	1
21	200604	ST_CLERK	50	1500	2500	1
22	200901	SH_CLERK	50	1500	3200	1
23	200502	SH_CLERK	50	1500	4100	1
24	200501	SA_MAN	80	2500	13500	1
25	200604	SA_REP	80	2500	8400	1
26	200701	ST_CLERK	50	1500	2400	1
27	200607	ST_CLERK	50	1500	2500	1
28	200602	SH_CLERK	50	1500	3100	1

--SalesOrderfactTemp_v2 + FACT

```
Create table salesOrderfactTemp_v2 as select
o.total_price, o.order_mode, oi.product_id, to_char(o.order_date, 'yyyymm') as orderTime_ID,
c.region_ID, o.order_date as transaction_date, p.start_date, p.end_date
From orders o, hosales.order_items oi, hosales.countries c, customers ct, hosales.promotions p
Where oi.order_id = o.order_id AND  o.customer_ID = ct.customer_ID AND ct.country_ID =
c.country_ID AND o.promotion_id = p.promotion_ID;

alter table salesOrderfactTemp_v2
add (season_Id number);

update salesOrderfactTemp_v2
set season_Id = 1
where to_char (transaction_date, 'mm')
in ('12','01','02');
```

```
update salesOrderfactTemp_v2
set season_Id = 2
where to_char (transaction_date, 'mm')
in ('03','04','05');

update salesOrderfactTemp_v2
set season_Id = 3
where to_char (transaction_date, 'mm')
in ('06','07','08');

update salesOrderfactTemp_v2
set season_Id = 4
where to_char (transaction_date, 'mm')
in ('09','10','11');

Create Table salesOrderfact_v2 as select
Order_mode, product_id, orderTime_ID, region_ID, season_ID,
Sum (total_price) as total_sales
From salesOrderfactTemp_v2
Where transaction_date >= start_date AND transaction_date <= end_date
Group by Order_mode, product_id, orderTime_ID, region_ID, season_ID;
```

	ORDER_MODE	PRODUCT_ID	ORDERTIME_ID	REGION_ID	SEASON_ID	TOTAL_SALES
1	online	2289	200806	2	3	115444.48
2	online	2289	200712	2	1	209503.77
3	direct	1910	200712	2	1	7338.52
4	online	2293	200706	2	3	236221.92
5	online	3165	200706	2	3	8891.52
6	direct	3127	200711	2	4	58381.38
7	direct	3133	200711	2	4	58381.38
8	direct	1948	200712	2	1	7338.52
9	direct	3150	200712	2	1	7824.67
10	online	2335	200712	2	1	209503.77
11	online	2350	200712	2	1	209503.77
12	online	1910	200806	2	3	100.8
13	direct	2810	200706	2	3	38.4
14	direct	3106	200711	2	4	58381.38
15	online	2276	200706	2	3	9106.48
16	online	3117	200806	2	3	47176
17	direct	3114	200711	2	4	12869.64
18	online	2311	200706	2	3	39951.12
19	online	2302	200712	2	1	209503.77
20	direct	3127	200712	2	1	7824.67
21	online	2311	200712	2	1	209503.77
22	direct	3139	200706	2	3	1140.8
23	direct	3143	200706	2	3	1140.8
24	online	3187	200706	2	3	8891.52
25	online	2326	200706	2	3	39951.12
26	online	2330	200706	2	3	257960.08
27	online	3193	200706	2	3	8891.52
28	direct	3170	200711	2	4	12869.64

Task 4 - OLAP Queries

(a). Reports with proper sub-totals:

REPORT 1: *"How many online orders in each season?" for each year using CUBE and DECODE*

Description: This query is useful to the management because it helps identify the popularity of their online system each year during every season. Using this they can see in general how popular their website is and which seasons are the most peak buy seasons, so they can target those specific seasons even more, by introducing sales etc.

SQL Query:

```
Select
    decode(years, NULL, 'All Years', years) as Years,
    decode(Season, NULL, 'All Seasons', Season) as Season, ONLINE_ORDERS from
(
Select
    to_char(to_date(orderTime_ID, 'yyyymm'), 'yyyy') as years,
    se.SeasonPeriod as Season,
    count(*) as ONLINE_ORDERS
from salesOrderFact_v1 s, SeasonDIM_v1 se
Where
    Order_mode = 'online'
and s.Season_ID = se.SeasonID
group by cube(to_char(to_date(orderTime_ID, 'yyyymm'), 'yyyy'), se.SeasonPeriod)
);
```

Output:

	YEARS	SEASON	ONLINE_ORDERS
1	All Years	All Seasons	57
2	All Years	Spring	43
3	All Years	Summer	14
4	2007	All Seasons	37
5	2007	Spring	23
6	2007	Summer	14
7	2008	All Seasons	20
8	2008	Spring	20

REPORT 2: *“What is the average sales for each region in each year?” for all different products using ROLLUP and DECODE.*

Description: This query is very useful to the management because it helps keep track of how every product has been performing in terms of its sales every year and the general trend over all years. Using this management can understand which products remained consistent meaning they are strong, and which did not and hence require improvements.

SQL Query:

```
Select DECODE(Year, NULL, 'All Years', Year) as Year, Region, Average_Sales
from(
Select
    to_char(to_date(s.orderTime_ID, 'yyyymm'), 'yyyy') as Year,
    DECODE(GROUPING(r.region_name), 1, 'All Regions', r.region_name) as Region,
    to_char(avg(s.total_sales), '$9,999,999,999') as Average_Sales
from salesOrderfact_V1 s, regionDim_V1 r
where s.region_ID = r.region_ID
group by ROLLUP(to_char(to_date(s.orderTime_ID, 'yyyymm'), 'yyyy'), r.region_name));
```

Output:

	YEAR	REGION	AVERAGE_SALES
1	2007	Americas	$43,596
2	2007	All Regions	$43,596
3	2008	Americas	$70,931
4	2008	All Regions	$70,931
5	All Years	All Regions	$49,063

(b). Reports with Rank and Percent_Rank:

REPORT 3: *“What is the Total Sales in each season (SeasonDIM) for each region (RegionDIM)?” Using RANK()*

Description:

This report clearly displays information that allows management to identify which regions are performing well, and which aren’t. This information is useful to see where their customer base is located.

SQL Query:

```
Select
    s.Season_ID as Seasons,
```

```
    r.region_name as Regions,
    to_char(sum(s.total_sales), '$9,999,999,999') as "TOTAL SALES",
    RANK() OVER (ORDER BY SUM(s.Total_Sales) DESC) AS "Rank"
from salesOrderfact_v1 s, RegionDIM_v1 r
Where s.region_id = r.region_id
Group By s.season_id, r.region_name;
```

Output:

	SEASONS	REGIONS	TOTAL SALES	Rank
1	3	Americas	$2,933,231.600	1
2	1	Americas	$1,460,533.130	2
3	4	Americas	$512,573.400	3

REPORT 4: *"Number of low-credit (CreditTypeDIM) customers in each region? (RegionDIM)" using PERCENT_RANK()*

Description:

This query allows management to higher interest regions by ranking the number of customers based on other regions.

SQL Query:

```
Select
    r.region_name as "Region Name",
    sum(c.No_Of_Customers) as "Number of Customers",
    percent_rank() over (order by sum(c.No_Of_Customers) desc) as "Percent Rank"
from Customerfact_v1 c, RegionDIM_v1 r, CountryDIM_v1 co
Where
    c.CreditType = 'High'
and c.Country_ID = co.Country_ID
and co.Region_ID = r.region_id
Group by r.region_name;
```

Output:

	Region Name	Number of Customers	Percent Rank
1	Americas	27	0
2	Asia	25	0.5
3	Europe	11	1

(c). Reports with Partitions:

Report 5: Total number of employees by department and job.

Description

This query is useful as it allows management to easily keep track of the sizes of departments. With the ranking by number of employees in each job, management can also keep track of which jobs have been allocated resources, and potentially which will need additional staff.

SQL Query:

```
select d.department_ID, j.job_title,
    Sum(e.number_of_employees) as total_employees,
    Rank() OVER (PARTITION BY d.department_ID order by
        sum(e.number_of_employees) DESC) As Rank
from departmentDIM_v1 d, Employeefact_v1 e, JobTypeDIM_v1 j
where d.department_ID = e.Department_ID and j.job_id = e.job_id
group by d.department_ID, j.job_title
order by d.department_ID;
```

Output:

	DEPARTMENT_ID	JOB_TITLE	TOTAL_EMPLOYEES	RANK
1	10	Administration Assistant	1	1
2	20	Marketing Representative	1	1
3	20	Marketing Manager	1	1
4	30	Purchasing Clerk	5	1
5	30	Purchasing Manager	1	2
6	40	Human Resources Representative	1	1
7	50	Stock Clerk	20	1
8	50	Shipping Clerk	20	1
9	50	Stock Manager	5	3
10	60	Programmer	5	1
11	70	Public Relations Representative	1	1
12	80	Sales Representative	30	1
13	80	Sales Manager	5	2
14	90	Administration Vice President	2	1
15	90	President	1	2
16	100	Accountant	5	1
17	100	Finance Manager	1	2
18	110	Public Accountant	1	1
19	110	Accounting Manager	1	1

Report 6: What is the total salary by each department and each job in each month?

Description:

This query allows management to easily identify where financial resources are being allocated between departments, and additionally, within each department. Rankings are provided for additional readability.

Query:

```
select d.department_ID, j.job_title,
    to_Char(Sum(e.Total_Salary), '$999,999.00') as Total_Salary_Expense,
    Rank() OVER (PARTITION BY d.department_ID order by
        sum(e.total_salary) DESC) As Rank
from departmentDIM_v1 d, Employeefact_v1 e, JobTypeDIM_v1 j
where d.department_ID = e.Department_ID and j.job_id = e.job_id
group by d.department_ID, j.job_title
order by d.department_ID;
```

Output:

	DEPARTMENT_ID	JOB_TITLE	TOTAL_SALARY_EXPENSE	RANK
1	10	Administration Assistant	$4,400.00	1
2	20	Marketing Manager	$13,000.00	1
3	20	Marketing Representative	$6,000.00	2
4	30	Purchasing Clerk	$13,900.00	1
5	30	Purchasing Manager	$11,000.00	2
6	40	Human Resources Representative	$6,500.00	1
7	50	Shipping Clerk	$64,300.00	1
8	50	Stock Clerk	$55,700.00	2
9	50	Stock Manager	$36,400.00	3
10	60	Programmer	$28,800.00	1
11	70	Public Relations Representative	$10,000.00	1
12	80	Sales Representative	$250,500.00	1
13	80	Sales Manager	$61,000.00	2
14	90	Administration Vice President	$34,000.00	1
15	90	President	$24,000.00	2
16	100	Accountant	$39,600.00	1
17	100	Finance Manager	$12,008.00	2
18	110	Accounting Manager	$12,008.00	1
19	110	Public Accountant	$9,300.00	2

(d). Reports with moving and cumulative aggregates:

REPORT 7: *"What is the total salary by each country and by each city?" using CUMULATIVE AGGREGATE and PARTITION*

Description:

This query is useful to the management because it helps in keeping track of the distribution of salary across various cities and countries. This is helpful to management because it tells which areas generate the most work or are very crucial to the business.

SQL Query:

```
Select
    l.city, c.country_name as COUNTRY,
    to_char(sum(e.total_salary), '$9,999,999,999') as "TOTAL SALARY",
    to_char(sum(sum(e.total_salary)) OVER
    (PARTITION BY c.country_name
    ORDER BY c.country_name, l.city
    ROWS UNBOUNDED PRECEDING),
    '$9,999,999,999') AS CUM_SALARY
from LocationDIM_v1 l, CountryDIM_v1 c, EmployeeFact_v1 e
Where
    e.Location_ID = l.Location_ID
and l.Country_ID = c.Country_ID
Group By l.city, c.country_name;
```

Output:

	CITY	COUNTRY	TOTAL SALARY	CUM_SALARY
1	Toronto	Canada	$19,000	$19,000
2	Munich	Germany	$10,000	$10,000
3	London	United Kingdom	$6,500	$6,500
4	Oxford	United Kingdom	$311,500	$318,000
5	Seattle	United States of America	$160,216	$160,216
6	South San Francisco	United States of America	$156,400	$316,616
7	Southlake	United States of America	$28,800	$345,416

REPORT 8: *"What is the total sales by each product and warehouses in each year/month" using MOVING AGGREGATE and PARTITION*

Description: This query is useful to the management because it shows how the sales are faring in the recent past for every product. This helps the management in seeing the current performance of their products and whether the results are as expected or they need to put more effort into some products to improve sales.

SQL Query:

```
Select
   p.product_name, w.warehouse_name, s.OrderTime_ID,
   to_char(sum(s.total_sales), '$9,999,999,999') as "TOTAL SALARY",
   to_char(sum(sum(s.total_sales)) OVER
   (PARTITION BY p.product_name
   ORDER BY p.product_name, w.warehouse_name
   ROWS 2 PRECEDING),
   '$9,999,999,999') AS MOVING_3_MONTH_SALES
from ProductDIM_v1 p, WarehouseDIM_v1 w, InventoryBridge_v1 i, SalesOrderfact_v1 s
Where
   s.Product_ID = p.Product_ID
and p.Product_ID = i.Product_ID
and i.Warehouse_ID = w.Warehouse_ID
Group By p.Product_Name, w.Warehouse_Name, s.OrderTime_ID;
```

Output:

All Output not shown

	PRODUCT_NAME	WAREHOUSE_NAME	ORDERTIME_ID	TOTAL SALARY	MOVING_3_MONTH_SALES
1	CPU D400	Beijing	200711	$8,208	$8,208
2	CPU D400	Bombay	200711	$8,208	$16,416
3	CPU D400	San Francisco	200711	$8,208	$24,624
4	CPU D400	Seattle, Washington	200711	$8,208	$24,624
5	CPU D400	Sydney	200711	$8,208	$24,624
6	Cable PR/S/6	Beijing	200711	$8,208	$8,208
7	Cable PR/S/6	Bombay	200711	$8,208	$16,416
8	Cable PR/S/6	Mexico City	200711	$8,208	$24,624
9	Cable PR/S/6	New Jersey	200711	$8,208	$24,624
10	Cable PR/S/6	San Francisco	200711	$8,208	$24,624
11	Cable PR/S/6	Seattle, Washington	200711	$8,208	$24,624
12	Cable PR/S/6	Southlake, Texas	200711	$8,208	$24,624
13	Cable PR/S/6	Sydney	200711	$8,208	$24,624
14	Cable PR/S/6	Toronto	200711	$8,208	$24,624
15	Chemicals - SW	Beijing	200712	$14,801	$14,801
16	Chemicals - SW	Bombay	200712	$14,801	$29,603
17	Chemicals - SW	Mexico City	200712	$14,801	$44,404
18	Chemicals - SW	New Jersey	200712	$14,801	$44,404
19	Chemicals - SW	Sydney	200712	$14,801	$44,404
20	Chemicals - SW	Toronto	200712	$14,801	$44,404

Task 5

Report 1

Original SQL

```
Select
    decode(years, NULL, 'All Years', years) as Years,
    decode(Season, NULL, 'All Seasons', Season) as Season, ONLINE_ORDERS from
(
Select
    to_char(to_date(orderTime_ID, 'yyyymm'), 'yyyy') as years,
    se.SeasonDesc as Season,
    count(*) as ONLINE_ORDERS
from salesOrderFact_v1 s, SeasonDIM_v1 se
Where
    Order_mode = 'online'
and s.Season_ID = se.Season_ID
group by cube(to_char(to_date(orderTime_ID, 'yyyymm'), 'yyyy'), se.SeasonDesc)
);
```

Original Query Result

	YEARS	SEASON	ONLINE_ORDERS
1	All Years	All Seasons	57
2	All Years	Spring	43
3	All Years	Summer	14
4	2007	All Seasons	37
5	2007	Spring	23
6	2007	Summer	14
7	2008	All Seasons	20
8	2008	Spring	20

Executed in: 0.069 Seconds

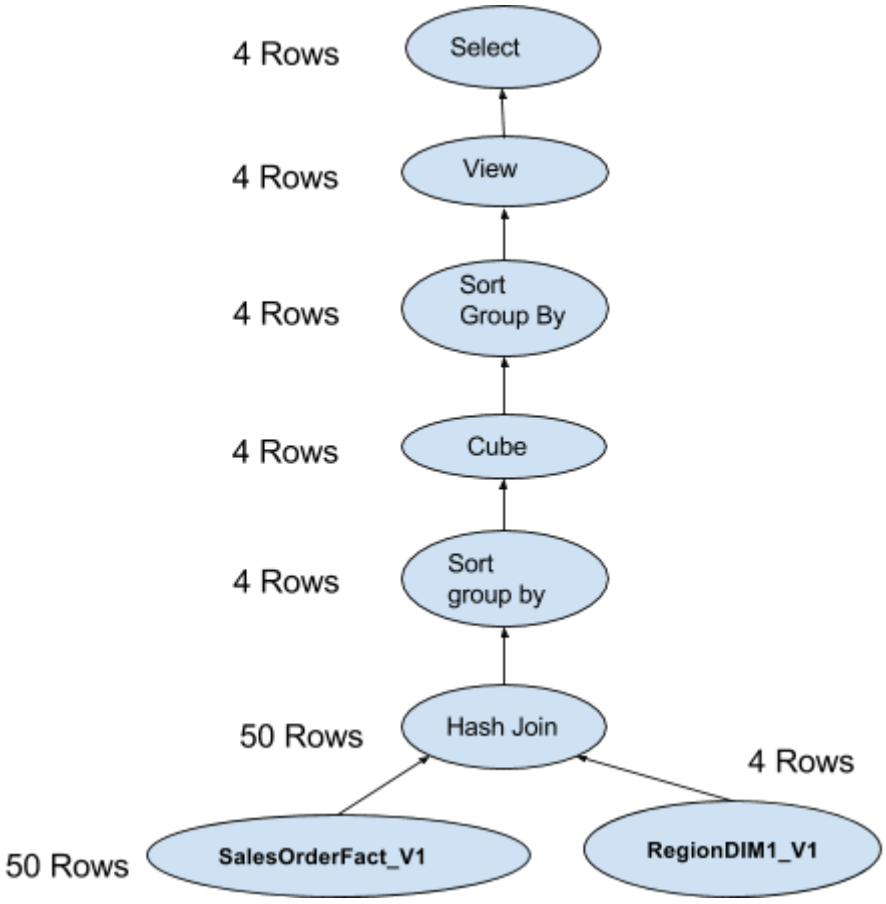
Original Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 430461560

-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                    |     4 |   116 |     7  (15)| 00:00:01 |
|   1 |  VIEW                  |                    |     4 |   116 |     7  (15)| 00:00:01 |
|   2 |   SORT GROUP BY        |                    |     4 |   168 |     7  (15)| 00:00:01 |
|   3 |    GENERATE CUBE       |                    |     4 |   168 |     7  (15)| 00:00:01 |
|   4 |     SORT GROUP BY      |                    |     4 |   168 |     7  (15)| 00:00:01 |
|*  5 |      HASH JOIN         |                    |    50 |  2100 |     6   (0)| 00:00:01 |
|   6 |       TABLE ACCESS FULL| SEASONDIM_V1       |     4 |   100 |     3   (0)| 00:00:01 |
|*  7 |       TABLE ACCESS FULL| SALESORDERFACT_V1  |    50 |   850 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Original Query Tree



New SQL:

```
Select
   decode(years, NULL, 'All Years', years) as Years,
   decode(Season, NULL, 'All Seasons', Season) as Season, ONLINE_ORDERS from
(
Select /*+ USE_MERGE (s se) */
   to_char(to_date(orderTime_ID, 'yyyymm'), 'yyyy') as years,
   se.SeasonDesc as Season,
   count(*) as ONLINE_ORDERS
from salesOrderFact_v1 s, SeasonDIM_v1 se
Where
   Order_mode = 'online'
and s.Season_ID = se.Season_ID
group by cube(to_char(to_date(orderTime_ID, 'yyyymm'), 'yyyy'), se.SeasonDesc)
);
```

New Query Result

	YEARS	SEASON	ONLINE_ORDERS
1	All Years	All Seasons	57
2	All Years	Spring	43
3	All Years	Summer	14
4	2007	All Seasons	37
5	2007	Spring	23
6	2007	Summer	14
7	2008	All Seasons	20
8	2008	Spring	20

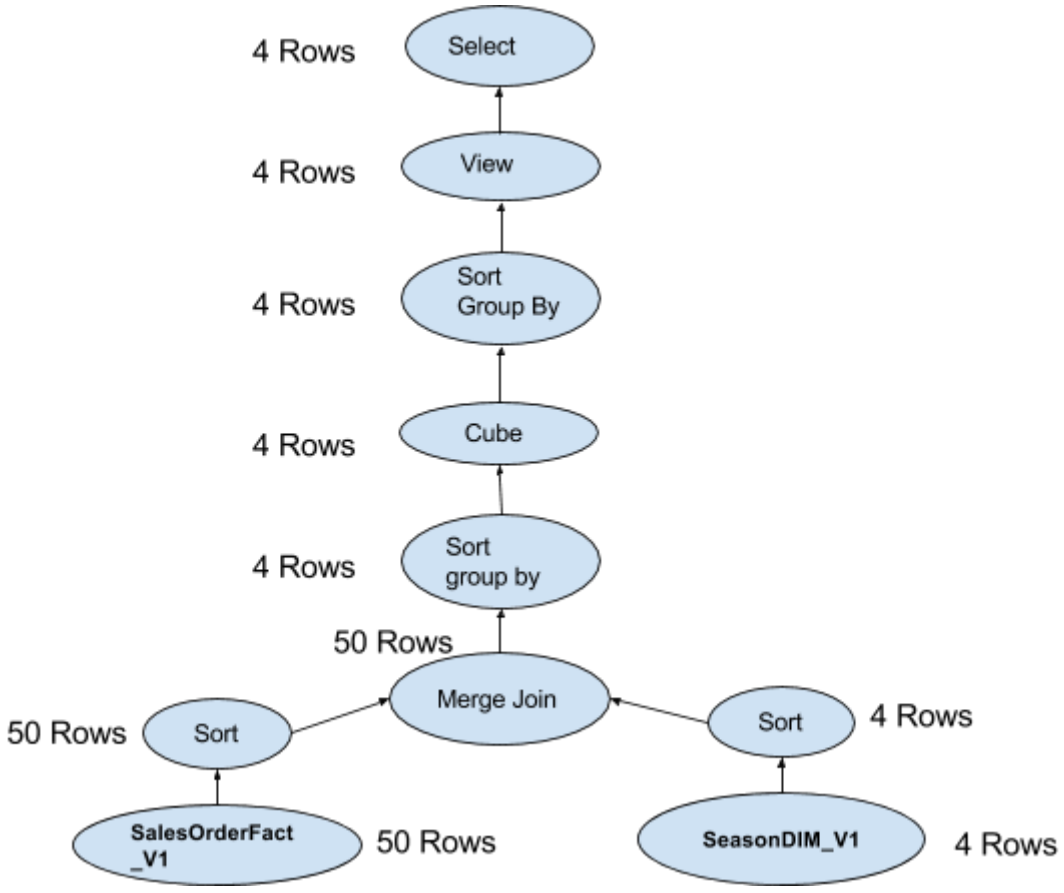
Execution Time: 0.109 seconds

New Execution Plan

```
PLAN_TABLE_OUTPUT
Plan hash value: 3799822551

-------------------------------------------------------------------------------------------------
| Id  | Operation                | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                    |     4 |   116 |     9  (34)| 00:00:01 |
|   1 |  VIEW                    |                    |     4 |   116 |     9  (34)| 00:00:01 |
|   2 |   SORT GROUP BY          |                    |     4 |   168 |     9  (34)| 00:00:01 |
|   3 |    GENERATE CUBE         |                    |     4 |   168 |     9  (34)| 00:00:01 |
|   4 |     SORT GROUP BY        |                    |     4 |   168 |     9  (34)| 00:00:01 |
|   5 |      MERGE JOIN          |                    |    50 |  2100 |     8  (25)| 00:00:01 |
|   6 |       SORT JOIN          |                    |     4 |   100 |     4  (25)| 00:00:01 |
|   7 |        TABLE ACCESS FULL| SEASONDIM_V1       |     4 |   100 |     3   (0)| 00:00:01 |
|*  8 |       SORT JOIN          |                    |    50 |   850 |     4  (25)| 00:00:01 |
|*  9 |        TABLE ACCESS FULL| SALESORDERFACT_V1  |    50 |   850 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

New Query Tree



Explanation
New query is less efficient because USE_MERGE sorts input tables and merges, HASH JOIN simply merges tables without sorting, by creating a hash for each row in one table and then joining with the other if hash matches.

Report 2

Original SQL:

```
Select DECODE(Year, NULL, 'All Years', Year) as Year, Region, Average_Sales
from(
Select
    to_char(to_date(s.orderTime_ID, 'yyyymm'), 'yyyy') as Year,
    DECODE(GROUPING(r.region_name), 1, 'All Regions', r.region_name) as Region,
    to_char(avg(s.total_sales), '$9,999,999,999') as Average_Sales
from salesOrderfact_V1 s, regionDim_V1 r
where s.region_ID = r.region_ID
group by ROLLUP(to_char(to_date(s.orderTime_ID, 'yyyymm'), 'yyyy'), r.region_name));
```

Original Query Result:

	YEAR	REGION	AVERAGE_SALES
1	2007	Americas	$43,596
2	2007	All Regions	$43,596
3	2008	Americas	$70,931
4	2008	All Regions	$70,931
5	All Years	All Regions	$49,063

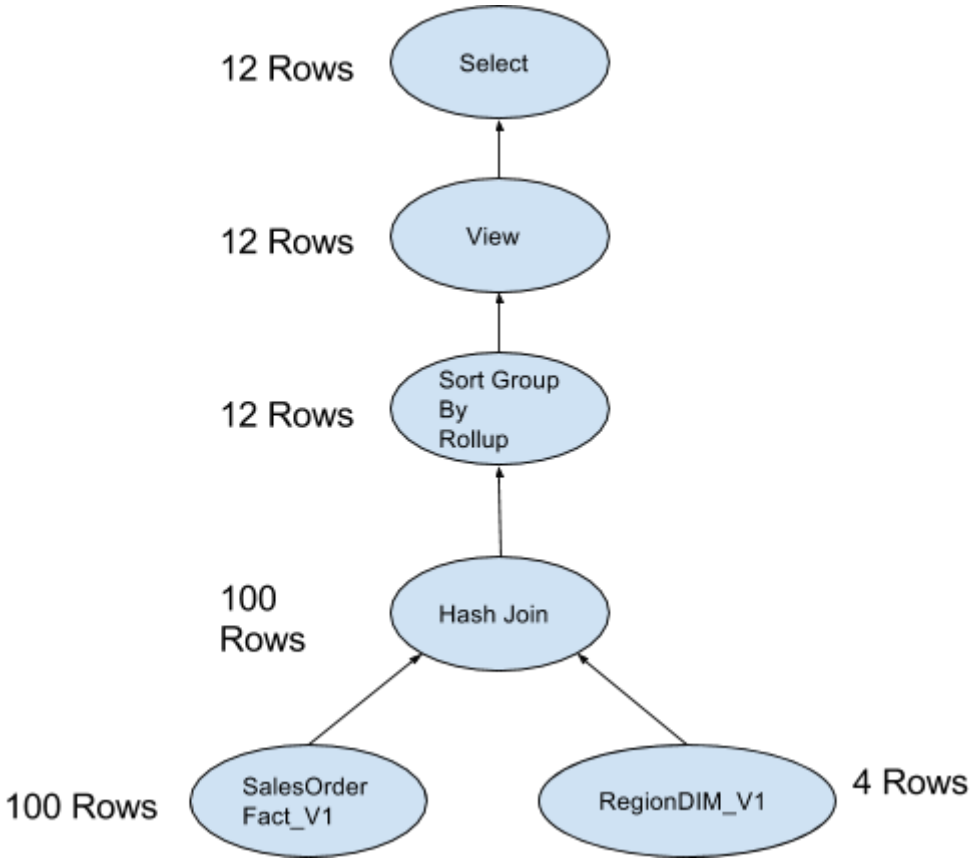
Execution Time: 0.056 seconds

Original Execution Plan:

```
PLAN_TABLE_OUTPUT
Plan hash value: 319777870

-------------------------------------------------------------------------------------------------
| Id  | Operation             | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                    |    12 |   324 |     7  (15)| 00:00:01 |
|   1 |  VIEW                 |                    |    12 |   324 |     7  (15)| 00:00:01 |
|   2 |   SORT GROUP BY ROLLUP|                    |    12 |   360 |     7  (15)| 00:00:01 |
|*  3 |    HASH JOIN          |                    |   100 |  3000 |     6   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL | REGIONDIM_V1       |     4 |    56 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL | SALESORDERFACT_V1  |   100 |  1600 |     3   (0)| 00:00:01 |
```

Original Query Tree:



New SQL:

```
Select DECODE(Year, NULL, 'All Years', Year) as Year, Region, Average_Sales
from(
Select /*+ USE_NL (s r)*/
    to_char(to_date(s.orderTime_ID, 'yyyymm'), 'yyyy') as Year,
    DECODE(GROUPING(r.region_name), 1, 'All Regions', r.region_name) as Region,
    to_char(avg(s.total_sales), '$9,999,999,999') as Average_Sales
from salesOrderfact_v1 s, regionDim_v1 r
where s.region_ID = r.region_ID
group by ROLLUP(to_char(to_date(s.orderTime_ID, 'yyyymm'), 'yyyy'), r.region_name));
```

New Query Result:

	YEAR	REGION	AVERAGE_SALES
1	2007	Americas	$43,596
2	2007	All Regions	$43,596
3	2008	Americas	$70,931
4	2008	All Regions	$70,931
5	All Years	All Regions	$49,063

Execution Time: 0.08 seconds

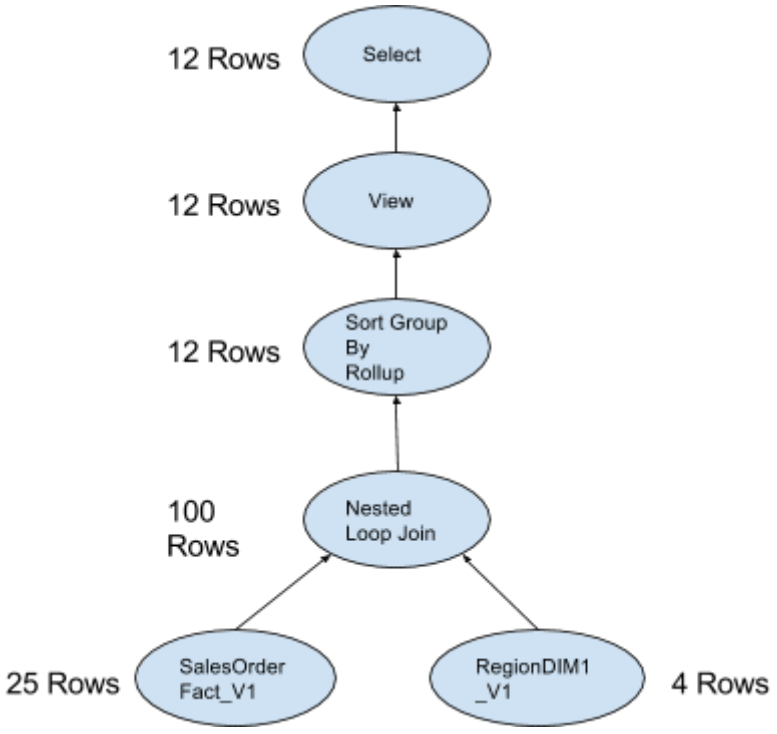
New Execution Plan:

PLAN_TABLE_OUTPUT

```
Plan hash value: 3900021802

-------------------------------------------------------------------------------------------------
| Id  | Operation             | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                    |    12 |   324 |    10  (10)| 00:00:01 |
|   1 |  VIEW                 |                    |    12 |   324 |    10  (10)| 00:00:01 |
|   2 |   SORT GROUP BY ROLLUP|                    |    12 |   360 |    10  (10)| 00:00:01 |
|   3 |    NESTED LOOPS       |                    |   100 |  3000 |     9   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL | REGIONDIM_V1       |     4 |    56 |     3   (0)| 00:00:01 |
|*  5 |     TABLE ACCESS FULL | SALESORDERFACT_V1  |    25 |   400 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

New Query Tree:



Explanation:
The new query is less efficient. Nested loop is takes more processing operations than hash join. Even though they have same execution time on average in this case, but nested loop join less efficient than Hash Join.

Report 3

Original SQL
Select

```
    s.Season_ID as Seasons,
    r.region_name as Regions,
    to_char(sum(s.total_sales), '$9,999,999,999'),
    RANK() OVER (ORDER BY SUM(s.Total_Sales) DESC) AS "Rank"
from salesOrderfact_v1 s, RegionDIM_v1 r
Where s.region_id = r.region_id
Group By s.season_id, r.region_name;
```

Original Query Result

	SEASONS	REGIONS	TOTAL SALES	Rank
1	3	Americas	$2,933,231.600	1
2	1	Americas	$1,460,533.130	2
3	4	Americas	$512,573.400	3

Executed in 0.052 Seconds

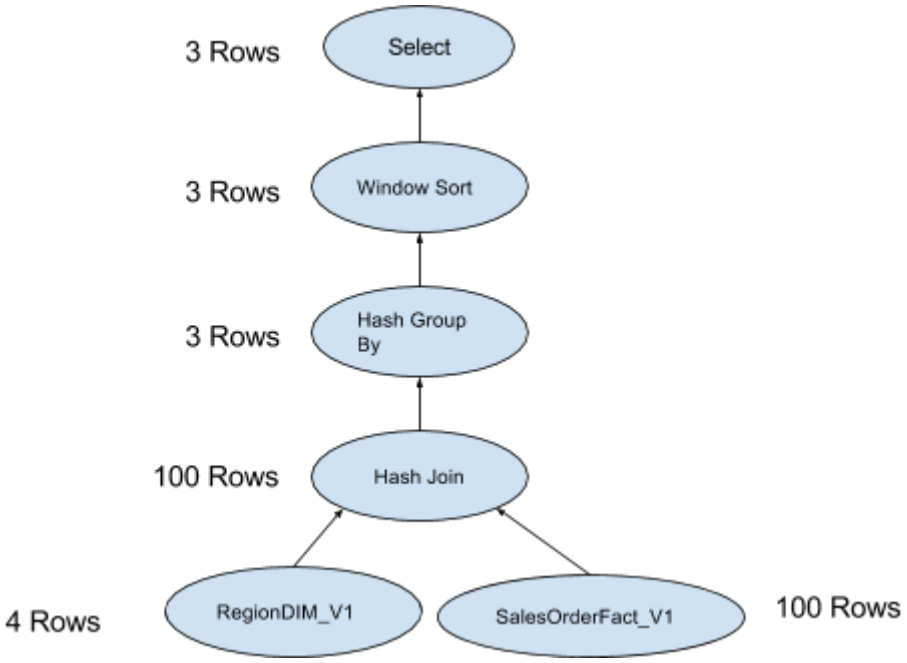
Original Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 1058959708

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                   |     3 |    78 |     8  (25)| 00:00:01 |
|   1 |  WINDOW SORT         |                   |     3 |    78 |     8  (25)| 00:00:01 |
|   2 |   HASH GROUP BY      |                   |     3 |    78 |     8  (25)| 00:00:01 |
|*  3 |    HASH JOIN         |                   |   100 |  2600 |     6   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| REGIONDIM_V1      |     4 |    56 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| SALESORDERFACT_V1 |   100 |  1200 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Original Query Tree



New SQL:

```
Select /*+ USE_MERGE (s r) */
    s.Season_ID as Seasons,
    r.region_name as Regions,
    to_char(sum(s.total_sales), '$9,999,999,999') as "Total Sales",
    RANK() OVER (ORDER BY SUM(s.Total_Sales) DESC) AS "Rank"
from salesOrderFact_v1 s, RegionDIM_v1 r
Where s.region_id = r.region_id
Group By s.season_id, r.region_name;
```

New Query Result

	SEASONS	REGIONS	TOTAL SALES	Rank
1	3	Americas	$2,933,231.600	1
2	1	Americas	$1,460,533.130	2
3	4	Americas	$512,573.400	3

Execution Time: 0.098 seconds

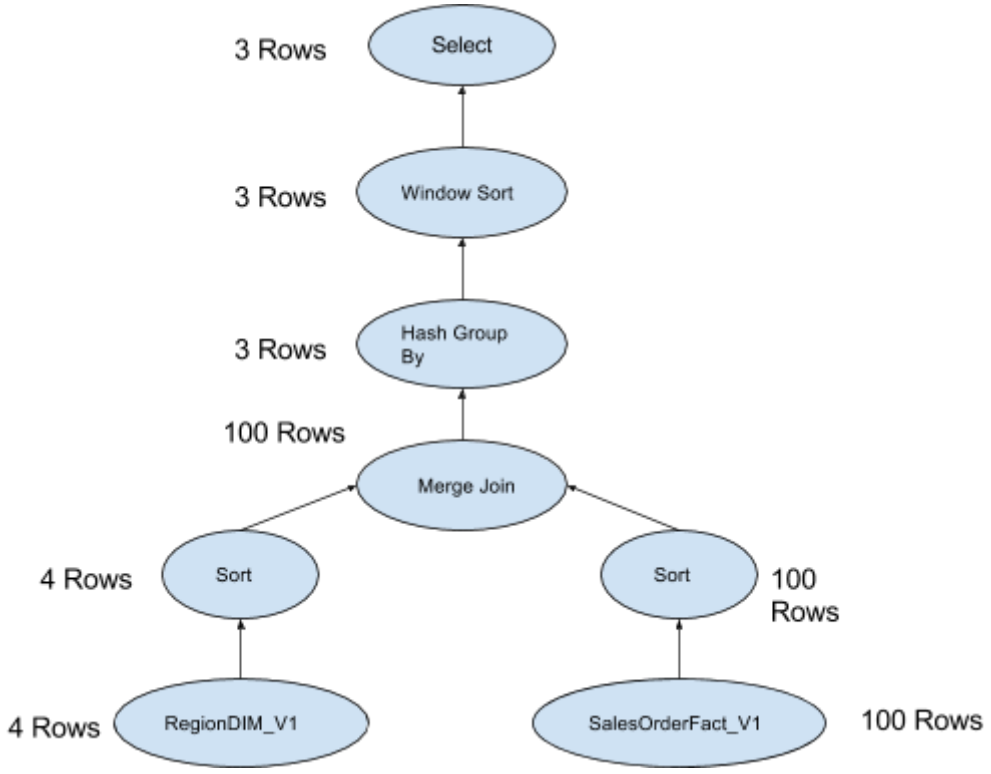
New Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 3357269700

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                    |     3 |    78 |    10  (40)| 00:00:01 |
|   1 |  WINDOW SORT          |                    |     3 |    78 |    10  (40)| 00:00:01 |
|   2 |   HASH GROUP BY       |                    |     3 |    78 |    10  (40)| 00:00:01 |
|   3 |    MERGE JOIN         |                    |   100 |  2600 |     8  (25)| 00:00:01 |
|   4 |     SORT JOIN         |                    |     4 |    56 |     4  (25)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| REGIONDIM_V1       |     4 |    56 |     3   (0)| 00:00:01 |
|*  6 |     SORT JOIN         |                    |   100 |  1200 |     4  (25)| 00:00:01 |
|   7 |      TABLE ACCESS FULL| SALESORDERFACT_V1  |   100 |  1200 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

New Query Tree



Explanation

New Query is less efficient because Merge Join needs to sort first but Hash Join does not. Sorting only happens in window sort for ordering by rank, when hash join is used. But for Merge Join, it sorts the input tables as well which is costly in terms of operations required and extra time needed. Hence New query is inefficient and first one is the more efficient one.

Report 4

Original SQL

```
Select
    r.region_name as "Region Name",
    sum(c.No_Of_Customers) as "Number of Customers",
    percent_rank() over (order by sum(c.No_Of_Customers) desc) as "Percent Rank"
from Customerfact_v1 c, RegionDIM_v1 r, CountryDIM_v1 co
Where
    c.CreditType = 'High'
and c.Country_ID = co.Country_ID
and co.Region_ID = r.region_id
Group by r.region_name;
```

Original Query Result

	Region Name	Number of Customers	Percent Rank
1	Americas	27	0
2	Asia	25	0.5
3	Europe	11	1

Executed in 0.029 Seconds

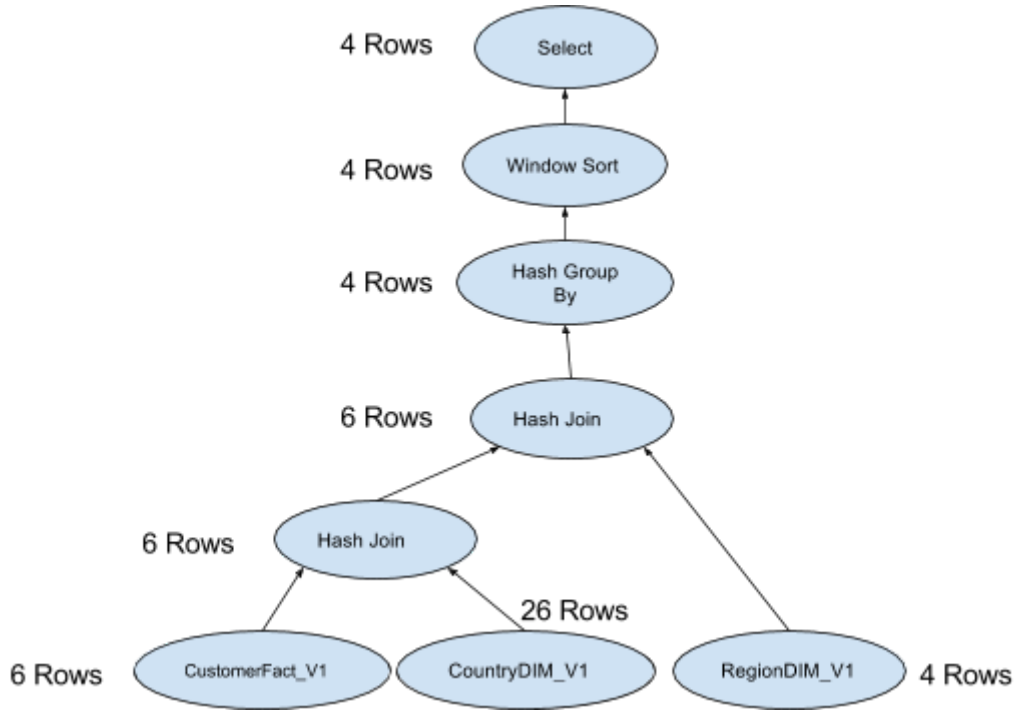
Original Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 2391721855

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                  |     4 |   120 |    11  (19)| 00:00:01 |
|   1 |  WINDOW SORT          |                  |     4 |   120 |    11  (19)| 00:00:01 |
|   2 |   HASH GROUP BY       |                  |     4 |   120 |    11  (19)| 00:00:01 |
|*  3 |    HASH JOIN          |                  |     6 |   180 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN         |                  |     6 |    96 |     6   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL| CUSTOMERFACT_V1  |     6 |    60 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| COUNTRYDIM_V1    |    26 |   156 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | REGIONDIM_V1     |     4 |    56 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Original Query Tree



New SQL:

```
Select
    r.region_name as "Region Name",
    sum(c.No_Of_Customers) as "Number of Customers",
    percent_rank() over (order by sum(c.No_Of_Customers) desc) as "Percent Rank"
from CustomerFact_v1 c, RegionDIM_v1 r, CountryDIM_v1 co
Where
    c.CreditType = 'High'
and c.Country_ID = co.Country_ID
and co.Region_ID = r.region_id
Group by r.region_name
order by sum(c.No_Of_Customers) desc;
```

New Query Result

	Region Name	Number of Customers	Percent Rank
1	Americas	27	0
2	Asia	25	0.5
3	Europe	11	1

Execution Time: 0.063 seconds

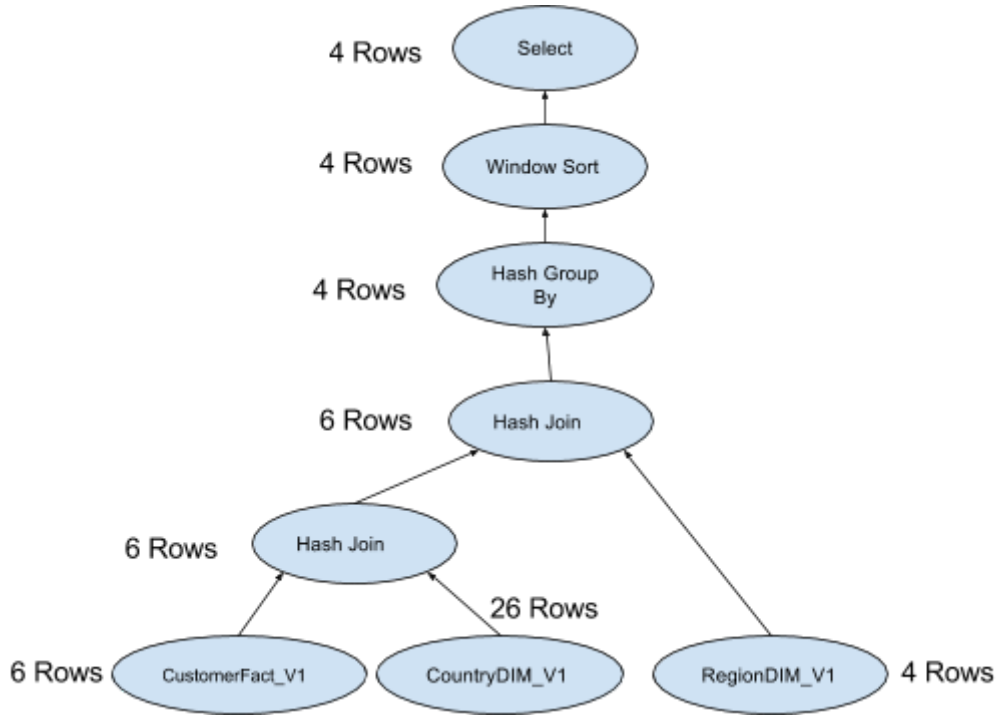
New Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 2391721855

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                 |     4 |   120 |    10  (10)| 00:00:01 |
|   1 |  WINDOW SORT         |                 |     4 |   120 |    10  (10)| 00:00:01 |
|   2 |   HASH GROUP BY      |                 |     4 |   120 |    10  (10)| 00:00:01 |
|*  3 |    HASH JOIN         |                 |     6 |   180 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN        |                 |     6 |    96 |     6   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL| CUSTOMERFACT_V1 |     6 |    60 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| COUNTRYDIM_V1   |    26 |   156 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | REGIONDIM_V1    |     4 |    56 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

New Query Tree



Explanation

New Query is less efficient because Order by at the end is redundant as if of no use. Makes the SQL query less efficient as a result. This is because percent_rank() does order by when calculating percent ranks. So system sees that and does not require another order by. If we add order by at the end, it is just waste and makes the query longer and inefficient.

Report 5

Original SQL

```
select d.department_ID, j.job_title,
    Sum(e.number_of_employees) as total_employees,
    Rank() OVER (PARTITION BY d.department_ID order by
        sum(e.number_of_employees) DESC) As Rank
from departmentDIM_v1 d, Employeefact_v1 e, JobTypeDIM_v1 j
where d.department_ID = e.Department_ID and j.job_id = e.job_id
group by d.department_ID, j.job_title
order by d.department_ID;
```

Original Query Result

	DEPARTMENT_ID	JOB_TITLE	TOTAL_EMPLOYEES	RANK
1	10	Administration Assistant	1	1
2	20	Marketing Representative	1	1
3	20	Marketing Manager	1	1
4	30	Purchasing Clerk	5	1
5	30	Purchasing Manager	1	2
6	40	Human Resources Representative	1	1
7	50	Stock Clerk	20	1
8	50	Shipping Clerk	20	1
9	50	Stock Manager	5	3
10	60	Programmer	5	1
11	70	Public Relations Representative	1	1
12	80	Sales Representative	30	1
13	80	Sales Manager	5	2
14	90	Administration Vice President	2	1
15	90	President	1	2
16	100	Accountant	5	1
17	100	Finance Manager	1	2
18	110	Public Accountant	1	1
19	110	Accounting Manager	1	1

Execute in 0.032 Seconds

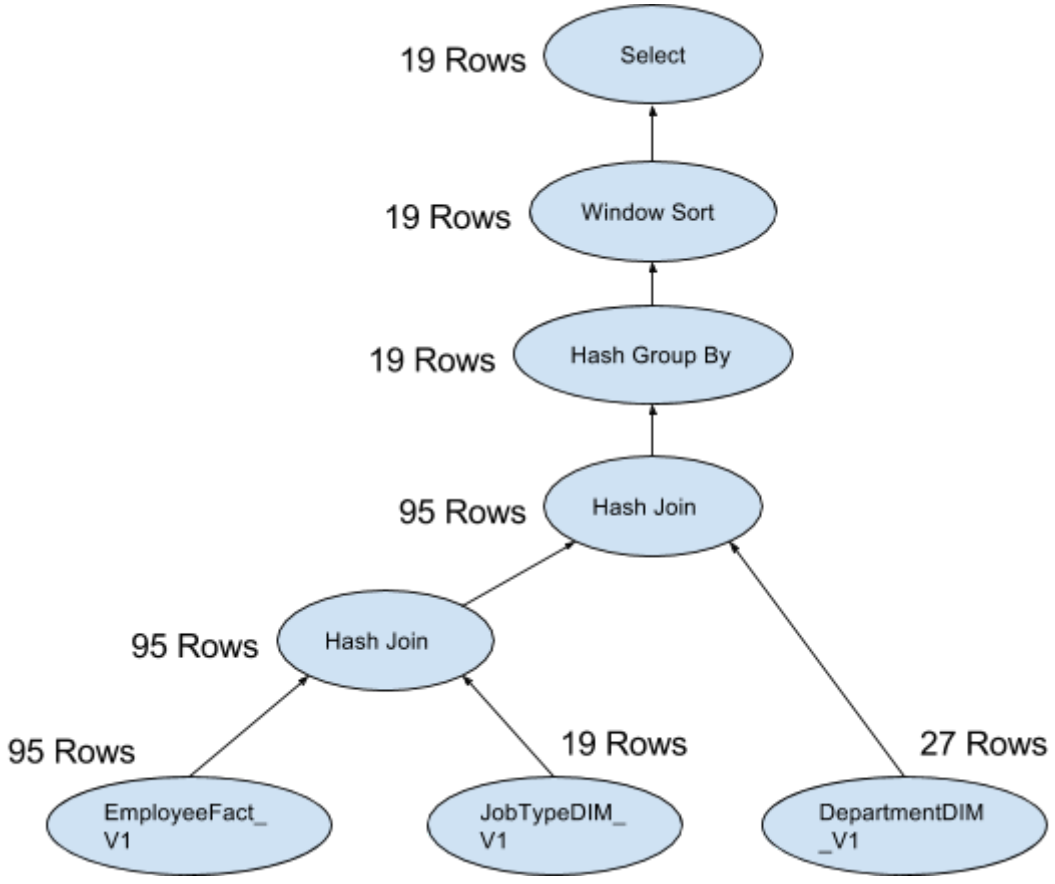
Original Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 3605420536

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                   |    19 |   893 |    10  (10)| 00:00:01 |
|   1 |  WINDOW SORT         |                   |    19 |   893 |    10  (10)| 00:00:01 |
|   2 |   HASH GROUP BY      |                   |    19 |   893 |    10  (10)| 00:00:01 |
|*  3 |    HASH JOIN         |                   |    95 |  4465 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN        |                   |    95 |  4085 |     6   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| JOBTYPEDIM_V1    |    19 |   513 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| EMPLOYEEFACT_V1  |    95 |  1520 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | DEPARTMENTDIM_V1 |    27 |   108 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Original Query Tree



New SQL:

```
Select /*+ ORDERED */ d.department_ID, j.job_title,
    Sum(e.number_of_employees) as total_employees,
    Rank() OVER (PARTITION BY d.department_ID order by
        sum(e.number_of_employees) DESC) As Rank
from departmentDIM_v1 d, JobTypeDim_v1 j, EmployeeFact_v1 e
where d.department_ID = e.Department_ID and j.job_id = e.job_id
```

group by d.department_ID, j.job_title
order by d.department_ID;

New Query Result

	DEPARTMENT_ID	JOB_TITLE	TOTAL_EMPLOYEES	RANK
1	10	Administration Assistant	1	1
2	20	Marketing Representative	1	1
3	20	Marketing Manager	1	1
4	30	Purchasing Clerk	5	1
5	30	Purchasing Manager	1	2
6	40	Human Resources Representative	1	1
7	50	Stock Clerk	20	1
8	50	Shipping Clerk	20	1
9	50	Stock Manager	5	3
10	60	Programmer	5	1
11	70	Public Relations Representative	1	1
12	80	Sales Representative	30	1
13	80	Sales Manager	5	2
14	90	Administration Vice President	2	1
15	90	President	1	2
16	100	Accountant	5	1
17	100	Finance Manager	1	2
18	110	Public Accountant	1	1
19	110	Accounting Manager	1	1

Execution Time: 0.08 seconds

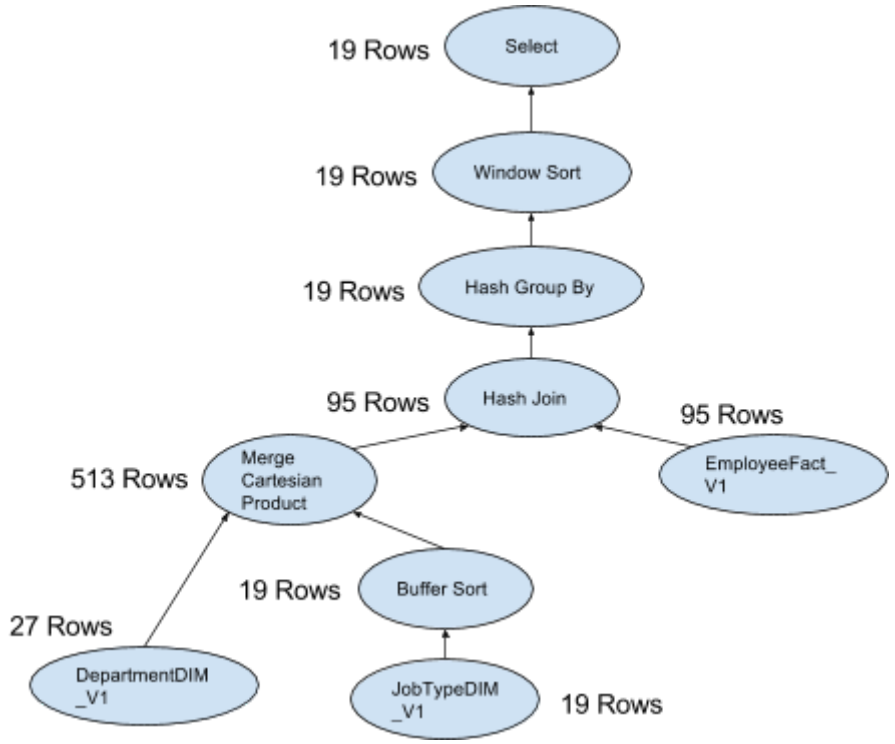
New Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 2039782011

-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                  |    19 |   893 |    38   (3)| 00:00:01 |
|   1 |  WINDOW SORT           |                  |    19 |   893 |    38   (3)| 00:00:01 |
|   2 |   HASH GROUP BY        |                  |    19 |   893 |    38   (3)| 00:00:01 |
|*  3 |    HASH JOIN           |                  |    95 |  4465 |    37   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL  | EMPLOYEEFACT_V1  |    95 |  1520 |     3   (0)| 00:00:01 |
|   5 |     MERGE JOIN CARTESIAN|                 |   513 | 15903 |    34   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL | DEPARTMENTDIM_V1 |    27 |   108 |     3   (0)| 00:00:01 |
|   7 |      BUFFER SORT       |                  |    19 |   513 |    31   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| JOBTYPEDIM_V1    |    19 |   513 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

New Query Tree



Explanation

New Query is less efficient because Fact size is not very big. So combining dimensions first using ORDERED does not not improve efficiency. But if the fact size increases in the future, then this will be more efficient.

Report 6

Original SQL

```
select d.department_ID, j.job_title,
    to_Char(Sum(e.Total_Salary), '$999,999.00') as Total_Salary_Expense,
    Rank() OVER (PARTITION BY d.department_ID order by
        sum(e.total_salary) DESC) As Rank
from departmentDIM_v1 d, Employeefact_v1 e, JobTypeDIM_v1 j
where d.department_ID = e.Department_ID and j.job_id = e.job_id
group by d.department_ID, j.job_title
order by d.department_ID;
```

Original Query Result

	DEPARTMENT_ID	JOB_TITLE	TOTAL_SALARY_EXPENSE	RANK
1	10	Administration Assistant	$4,400.00	1
2	20	Marketing Manager	$13,000.00	1
3	20	Marketing Representative	$6,000.00	2
4	30	Purchasing Clerk	$13,900.00	1
5	30	Purchasing Manager	$11,000.00	2
6	40	Human Resources Representative	$6,500.00	1
7	50	Shipping Clerk	$64,300.00	1
8	50	Stock Clerk	$55,700.00	2
9	50	Stock Manager	$36,400.00	3
10	60	Programmer	$28,800.00	1
11	70	Public Relations Representative	$10,000.00	1
12	80	Sales Representative	$250,500.00	1
13	80	Sales Manager	$61,000.00	2
14	90	Administration Vice President	$34,000.00	1
15	90	President	$24,000.00	2
16	100	Accountant	$39,600.00	1
17	100	Finance Manager	$12,008.00	2
18	110	Accounting Manager	$12,008.00	1
19	110	Public Accountant	$9,300.00	2

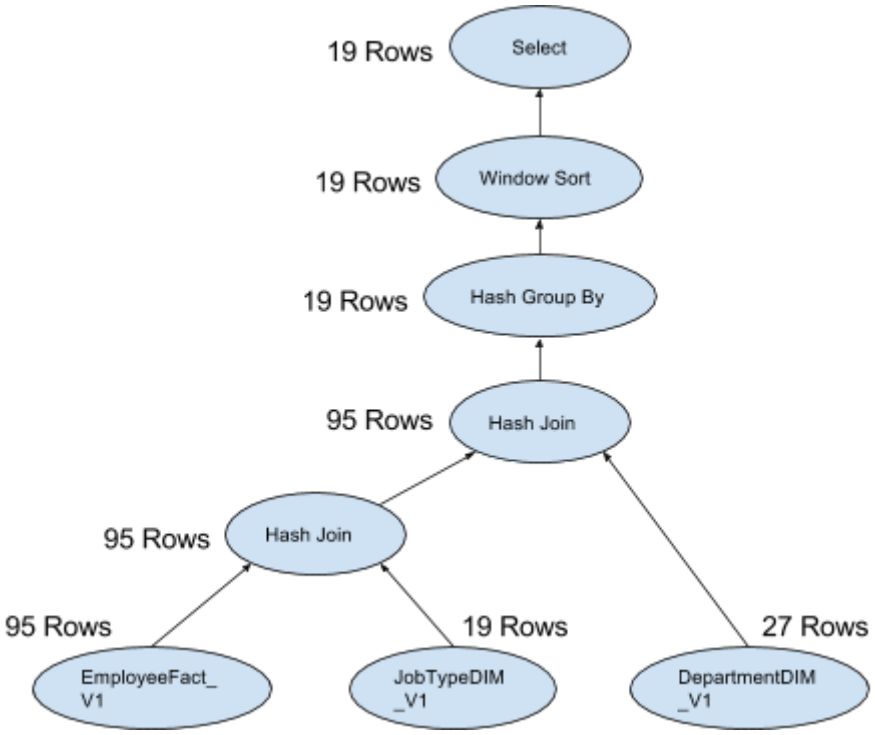
Executed in 0.027 Seconds

Original Execution Plan

```
PLAN_TABLE_OUTPUT
Plan hash value: 3605420536

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                   |    19 |   912 |    10  (10)| 00:00:01 |
|   1 |  WINDOW SORT          |                   |    19 |   912 |    10  (10)| 00:00:01 |
|   2 |   HASH GROUP BY       |                   |    19 |   912 |    10  (10)| 00:00:01 |
|*  3 |    HASH JOIN          |                   |    95 |  4560 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN         |                   |    95 |  4180 |     6   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| JOBTYPEDIM_V1     |    19 |   513 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| EMPLOYEEFACT_V1   |    95 |  1615 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | DEPARTMENTDIM_V1  |    27 |   108 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Original Query Tree



New SQL:

```
select /*+ USE_NL (d j)*/ d.department_ID, j.job_title,
    to_Char(Sum(e.Total_Salary), '$999,999.00') as Total_Salary_Expense,
    Rank() OVER (PARTITION BY d.department_ID order by
        sum(e.total_salary) DESC) As Rank
from departmentDIM_v1 d, EmployeeFact_v1 e, JobTypeDim_v1 j
where d.department_ID = e.Department_ID and j.job_id = e.job_id
group by d.department_ID, j.job_title
order by d.department_ID;
```

New Query Result

	DEPARTMENT_ID	JOB_TITLE	TOTAL_SALARY_EXPENSE	RANK
1	10	Administration Assistant	$4,400.00	1
2	20	Marketing Manager	$13,000.00	1
3	20	Marketing Representative	$6,000.00	2
4	30	Purchasing Clerk	$13,900.00	1
5	30	Purchasing Manager	$11,000.00	2
6	40	Human Resources Representative	$6,500.00	1
7	50	Shipping Clerk	$64,300.00	1
8	50	Stock Clerk	$55,700.00	2
9	50	Stock Manager	$36,400.00	3
10	60	Programmer	$28,800.00	1
11	70	Public Relations Representative	$10,000.00	1
12	80	Sales Representative	$250,500.00	1
13	80	Sales Manager	$61,000.00	2
14	90	Administration Vice President	$34,000.00	1
15	90	President	$24,000.00	2
16	100	Accountant	$39,600.00	1
17	100	Finance Manager	$12,008.00	2
18	110	Accounting Manager	$12,008.00	1
19	110	Public Accountant	$9,300.00	2

Execution Time: 0.084 seconds

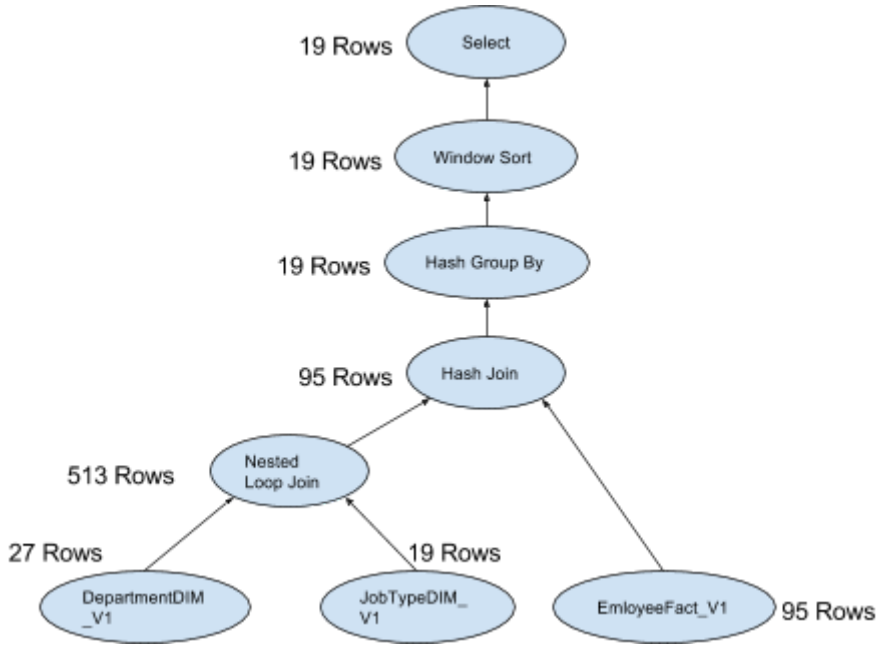
New Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 9272461

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                   |    19 |   912 |    29   (4)| 00:00:01 |
|   1 |  WINDOW SORT          |                   |    19 |   912 |    29   (4)| 00:00:01 |
|   2 |   HASH GROUP BY       |                   |    19 |   912 |    29   (4)| 00:00:01 |
|*  3 |    HASH JOIN          |                   |    95 |  4560 |    28   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL | EMPLOYEEFACT_V1   |    95 |  1615 |     3   (0)| 00:00:01 |
|   5 |     NESTED LOOPS      |                   |   513 | 15903 |    25   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| JOBTYPEDIM_V1     |    19 |   513 |     3   (0)| 00:00:01 |
|   7 |      TABLE ACCESS FULL| DEPARTMENTDIM_V1  |    27 |   108 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

New Query Tree



Explanation
New query is less efficient because fact size is not too big, so cartesian product of dimensions before fact does not increase efficiency. In this case it creates more number of rows to be carried in combining dimensions first than in the original query.

Report 7

Original SQL

```
Select
    l.city, c.country_name as COUNTRY,
    to_char(sum(e.total_salary), '$9,999,999,999') as "TOTAL SALARY",
    to_char(sum(sum(e.total_salary)) OVER
    (PARTITION BY c.country_name
    ORDER BY c.country_name, l.city
    ROWS UNBOUNDED PRECEDING),
    '$9,999,999,999') AS CUM_SALARY
from LocationDIM_v1 l, CountryDIM_v1 c, EmployeeFact_v1 e
Where
    e.Location_ID = l.Location_ID
and l.Country_ID = c.Country_ID
Group By l.city, c.country_name;
```

Original Query Result

	CITY	COUNTRY	TOTAL SALARY	CUM_SALARY
1	Toronto	Canada	$19,000	$19,000
2	Munich	Germany	$10,000	$10,000
3	London	United Kingdom	$6,500	$6,500
4	Oxford	United Kingdom	$311,500	$318,000
5	Seattle	United States of America	$160,216	$160,216
6	South San Francisco	United States of America	$156,400	$316,616
7	Southlake	United States of America	$28,800	$345,416

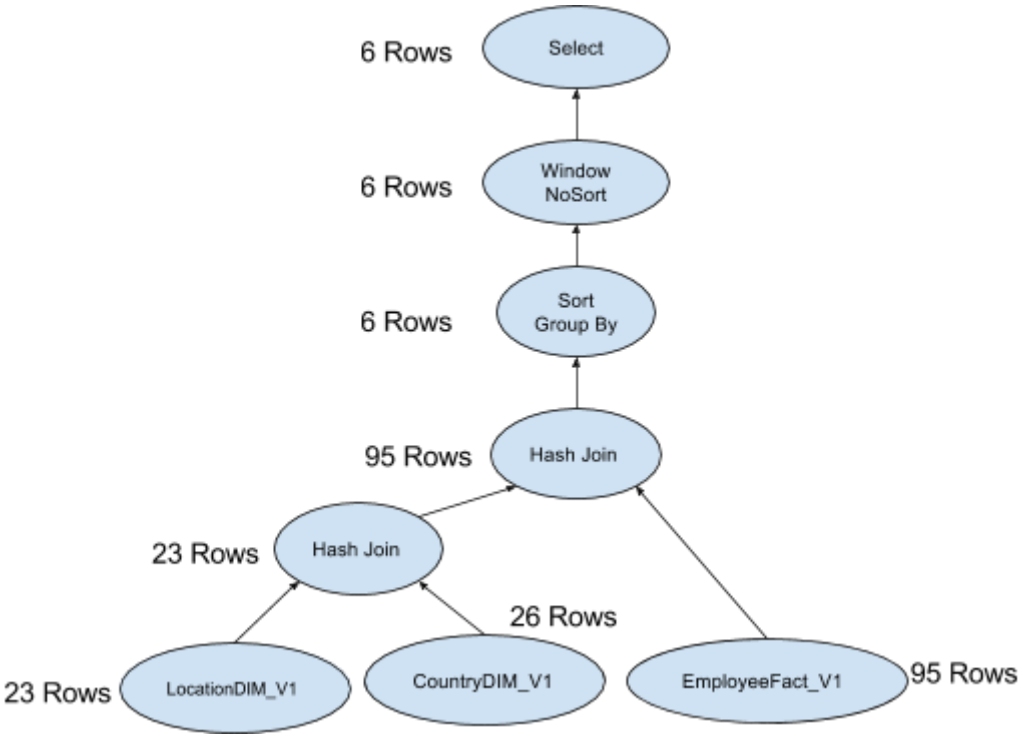
Executed in 0.032 Seconds

Original Execution Plan

```
PLAN_TABLE_OUTPUT
Plan hash value: 4199427109

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                 |     6 |   237 |    10  (10)| 00:00:01 |
|   1 |  WINDOW NOSORT        |                 |     6 |   237 |    10  (10)| 00:00:01 |
|   2 |   SORT GROUP BY       |                 |     6 |   237 |    10  (10)| 00:00:01 |
|*  3 |    HASH JOIN          |                 |    95 |  3230 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN         |                 |    23 |   621 |     6   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| LOCATIONDIM_V1  |    23 |   345 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| COUNTRYDIM_V1   |    26 |   312 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | EMPLOYEEFACT_V1 |    95 |   665 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Original Query Tree



New SQL

```
Select
   InnerQuery.city, InnerQuery.country_name,
   InnerQuery.total_salary as "TOTAL SALARY",
   to_char(sum(InnerQuery.total_salary) OVER
   (PARTITION BY InnerQuery.country_name
   ORDER BY InnerQuery.country_name, InnerQuery.city
   ROWS UNBOUNDED PRECEDING),
   '$9,999,999,999.99') AS CUM_SALARY
from
((Select /*+ no_merge */
   l.city as city, c.country_name as country_name, sum(e.total_salary) as total_salary
from LocationDIM_v1 l, CountryDIM_v1 c, EmployeeFact_v1 e
Where
   e.Location_ID = l.Location_ID
and l.Country_ID = c.Country_ID
Group By l.city, c.country_name) InnerQuery);
```

New Query Result

	CITY	COUNTRY	TOTAL SALARY	CUM_SALARY
1	Toronto	Canada	$19,000	$19,000
2	Munich	Germany	$10,000	$10,000
3	London	United Kingdom	$6,500	$6,500
4	Oxford	United Kingdom	$311,500	$318,000
5	Seattle	United States of America	$160,216	$160,216
6	South San Francisco	United States of America	$156,400	$316,616
7	Southlake	United States of America	$28,800	$345,416

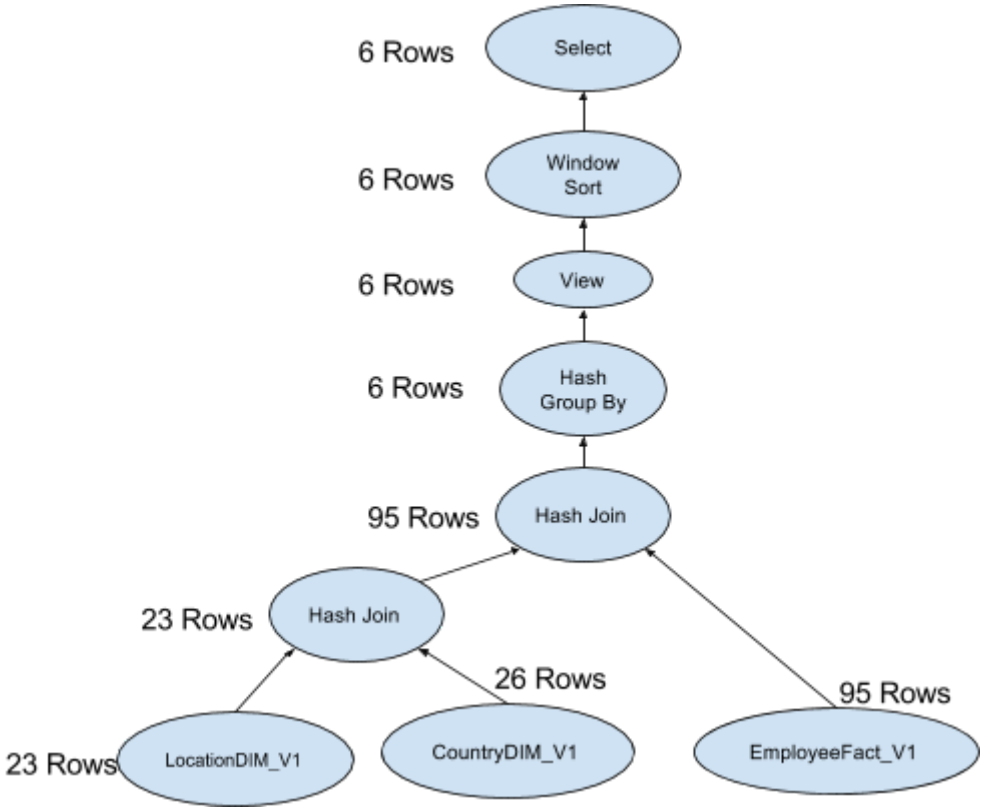
Execution Time: 0.025 seconds

New Execution Plan

```
PLAN_TABLE_OUTPUT
Plan hash value: 63626029

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                  |     6 |   216 |    11  (19)| 00:00:01 |
|   1 |  WINDOW SORT          |                  |     6 |   216 |    11  (19)| 00:00:01 |
|   2 |   VIEW                |                  |     6 |   216 |    10  (10)| 00:00:01 |
|   3 |    HASH GROUP BY      |                  |     6 |   237 |    10  (10)| 00:00:01 |
|*  4 |     HASH JOIN         |                  |    95 |  3230 |     9   (0)| 00:00:01 |
|*  5 |      HASH JOIN        |                  |    23 |   621 |     6   (0)| 00:00:01 |
|   6 |       TABLE ACCESS FULL| LOCATIONDIM_V1  |    23 |   345 |     3   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| COUNTRYDIM_V1   |    26 |   312 |     3   (0)| 00:00:01 |
|   8 |      TABLE ACCESS FULL | EMPLOYEEFACT_V1 |    95 |   665 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

New Query Tree



Explanation

The New Query is more efficient because sorting happens once tables are grouped not on the input tables directly. In the original query, tables are sorted and grouped during this sorting which is not very efficient if the tables are very large. With the new query, tables are grouped first and then sorted, making sorting much more efficient because grouped tables are much smaller than normal table.

Report 8

Original SQL

```
Select
   p.product_name, w.warehouse_name, s.OrderTime_ID,
   to_char(sum(s.total_sales), '$9,999,999,999') as "TOTAL SALARY",
   to_char(sum(sum(s.total_sales)) OVER
   (PARTITION BY p.product_name
   ORDER BY p.product_name, w.warehouse_name
   ROWS 2 PRECEDING),
   '$9,999,999,999') AS MOVING_3_MONTH_SALES
from ProductDIM_v1 p, WarehouseDIM_v1 w, InventoryBridge_v1 i, SalesOrderfact_v1 s
Where
   s.Product_ID = p.Product_ID
and p.Product_ID = i.Product_ID
```

and i.Warehouse_ID = w.Warehouse_ID
Group By p.Product_Name, w.Warehouse_Name, s.OrderTime_ID;

Original Query Result (All output not shown)

	PRODUCT_NAME	WAREHOUSE_NAME	ORDERTIME_ID	TOTAL SALARY	MOVING_3_MONTH_SALES
1	CPU D400	Beijing	200711	$8,208	$8,208
2	CPU D400	Bombay	200711	$8,208	$16,416
3	CPU D400	San Francisco	200711	$8,208	$24,624
4	CPU D400	Seattle, Washington	200711	$8,208	$24,624
5	CPU D400	Sydney	200711	$8,208	$24,624
6	Cable PR/S/6	Beijing	200711	$8,208	$8,208
7	Cable PR/S/6	Bombay	200711	$8,208	$16,416
8	Cable PR/S/6	Mexico City	200711	$8,208	$24,624
9	Cable PR/S/6	New Jersey	200711	$8,208	$24,624
10	Cable PR/S/6	San Francisco	200711	$8,208	$24,624
11	Cable PR/S/6	Seattle, Washington	200711	$8,208	$24,624
12	Cable PR/S/6	Southlake, Texas	200711	$8,208	$24,624
13	Cable PR/S/6	Sydney	200711	$8,208	$24,624
14	Cable PR/S/6	Toronto	200711	$8,208	$24,624
15	Chemicals - SW	Beijing	200712	$14,801	$14,801
16	Chemicals - SW	Bombay	200712	$14,801	$29,603
17	Chemicals - SW	Mexico City	200712	$14,801	$44,404
18	Chemicals - SW	New Jersey	200712	$14,801	$44,404
19	Chemicals - SW	Sydney	200712	$14,801	$44,404
20	Chemicals - SW	Toronto	200712	$14,801	$44,404

Executed in 0.073 Seconds

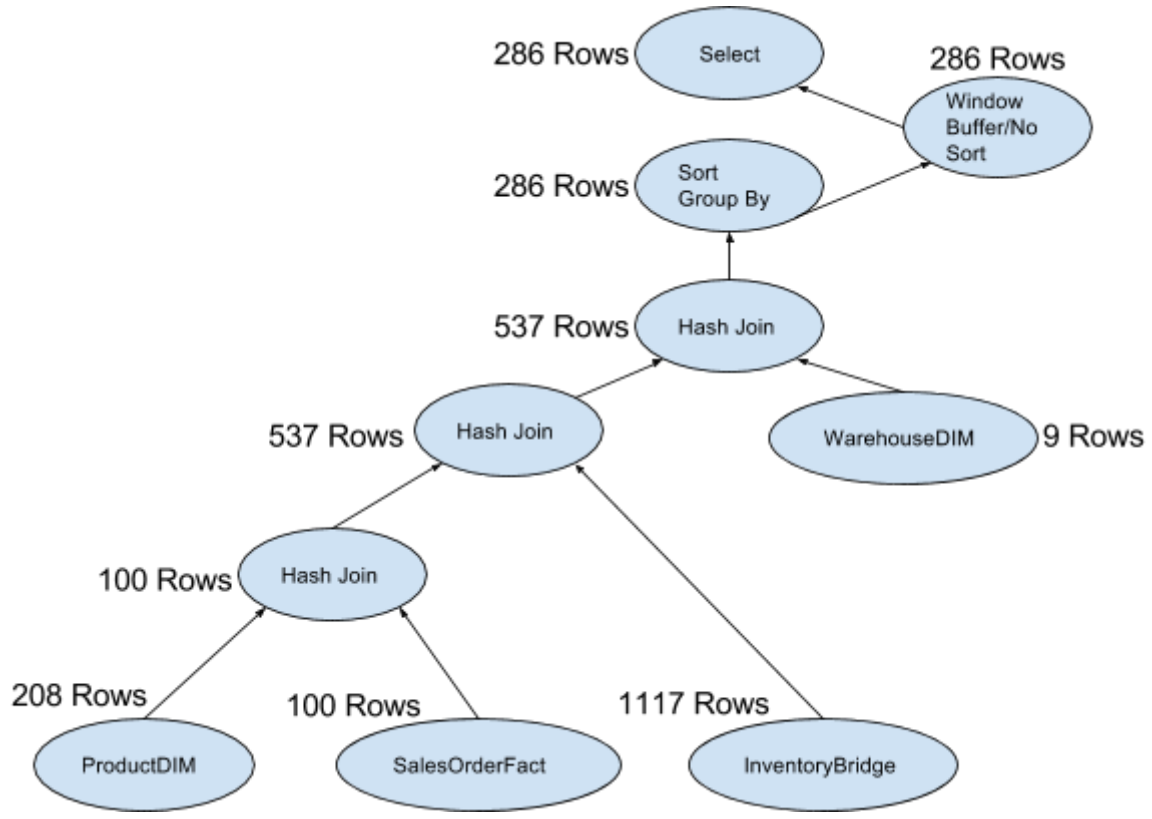
Original Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 4033695777

-----------------------------------------------------------------------------------------------
| Id  | Operation             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                     |   286 | 17160 |    14   (8)| 00:00:01 |
|   1 |  WINDOW BUFFER        |                     |   286 | 17160 |    14   (8)| 00:00:01 |
|   2 |   SORT GROUP BY       |                     |   286 | 17160 |    14   (8)| 00:00:01 |
|*  3 |    HASH JOIN          |                     |   537 | 32220 |    13   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL | WAREHOUSEDIM_V1     |     9 |   135 |     3   (0)| 00:00:01 |
|*  5 |     HASH JOIN         |                     |   537 | 24165 |    10   (0)| 00:00:01 |
|*  6 |      HASH JOIN        |                     |   100 |  3800 |     7   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| SALESORDERFACT_V1  |   100 |  1700 |     3   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| PRODUCTDIM_V1      |   208 |  4368 |     4   (0)| 00:00:01 |
|   9 |      TABLE ACCESS FULL | INVENTORYBRIDGE_V1 |  1117 |  7819 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Original Query Tree



New SQL

```
Select
   InnerQuery.product_name, InnerQuery.warehouse_name, InnerQuery.OrderTime_ID,
   to_char(InnerQuery.total_sales, '$9,999,999,999') as "TOTAL SALARY",
   to_char(sum(InnerQuery.total_sales) OVER
   (PARTITION BY InnerQuery.product_name
```

```
    ORDER BY InnerQuery.product_name, InnerQuery.warehouse_name
    ROWS 2 PRECEDING),
    '$9,999,999,999') AS MOVING_3_MONTH_SALES
from
((Select /*+ no_merge */
    p.product_name as product_name, w.warehouse_name as warehouse_name,
    s.orderTime_ID as OrderTime_ID, sum(s.total_sales) as total_sales
from ProductDIM_v1 p, WarehouseDIM_v1 w, InventoryBridge_v1 i, SalesOrderFact_v1 s
Where
    s.Product_ID = p.Product_ID
and p.Product_ID = i.Product_ID
and i.Warehouse_ID = w.Warehouse_ID
Group By p.Product_Name, w.Warehouse_Name, s.OrderTime_ID) InnerQuery);
```

New Query Result (All Output not shown)

	PRODUCT_NAME	WAREHOUSE_NAME	ORDERTIME_ID	TOTAL SALARY	MOVING_3_MONTH_SALES
1	CPU D400	Beijing	200711	$8,208	$8,208
2	CPU D400	Bombay	200711	$8,208	$16,416
3	CPU D400	San Francisco	200711	$8,208	$24,624
4	CPU D400	Seattle, Washington	200711	$8,208	$24,624
5	CPU D400	Sydney	200711	$8,208	$24,624
6	Cable PR/S/6	Beijing	200711	$8,208	$8,208
7	Cable PR/S/6	Bombay	200711	$8,208	$16,416
8	Cable PR/S/6	Mexico City	200711	$8,208	$24,624
9	Cable PR/S/6	New Jersey	200711	$8,208	$24,624
10	Cable PR/S/6	San Francisco	200711	$8,208	$24,624
11	Cable PR/S/6	Seattle, Washington	200711	$8,208	$24,624
12	Cable PR/S/6	Southlake, Texas	200711	$8,208	$24,624
13	Cable PR/S/6	Sydney	200711	$8,208	$24,624
14	Cable PR/S/6	Toronto	200711	$8,208	$24,624
15	Chemicals - SW	Beijing	200712	$14,801	$14,801
16	Chemicals - SW	Bombay	200712	$14,801	$29,603
17	Chemicals - SW	Mexico City	200712	$14,801	$44,404
18	Chemicals - SW	New Jersey	200712	$14,801	$44,404
19	Chemicals - SW	Sydney	200712	$14,801	$44,404
20	Chemicals - SW	Toronto	200712	$14,801	$44,404

Execution Time: 0.062 seconds

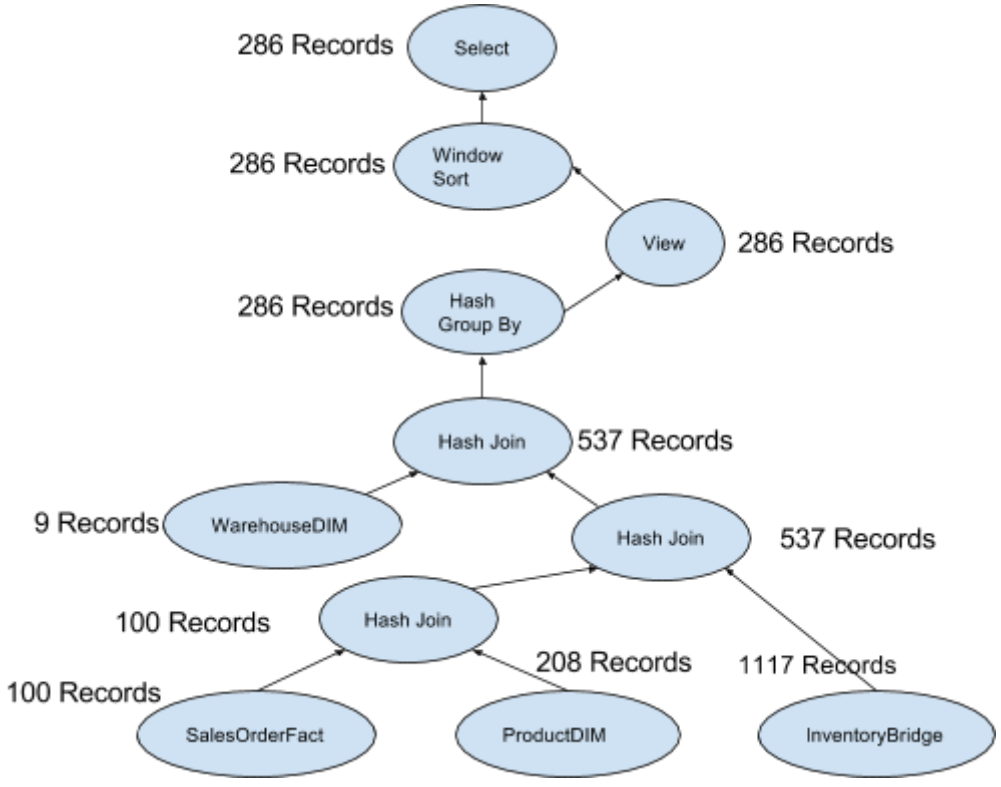
New Execution Plan

PLAN_TABLE_OUTPUT

```
Plan hash value: 624303534

-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                     |   286 | 13442 |    14   (8)| 00:00:01 |
|   1 |  WINDOW SORT             |                     |   286 | 13442 |    14   (8)| 00:00:01 |
|   2 |   VIEW                   |                     |   286 | 13442 |    14   (8)| 00:00:01 |
|   3 |    HASH GROUP BY         |                     |   286 | 17160 |    14   (8)| 00:00:01 |
|*  4 |     HASH JOIN            |                     |   537 | 32220 |    13   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL   | WAREHOUSEDIM_V1     |     9 |   135 |     3   (0)| 00:00:01 |
|*  6 |      HASH JOIN           |                     |   537 | 24165 |    10   (0)| 00:00:01 |
|*  7 |       HASH JOIN          |                     |   100 |  3800 |     7   (0)| 00:00:01 |
|   8 |        TABLE ACCESS FULL| SALESORDERFACT_V1   |   100 |  1700 |     3   (0)| 00:00:01 |
|   9 |        TABLE ACCESS FULL| PRODUCTDIM_V1       |   208 |  4368 |     4   (0)| 00:00:01 |
|  10 |       TABLE ACCESS FULL | INVENTORYBRIDGE_V1  |  1117 |  7819 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

New Query Tree



Explanation

The New Query is more efficient because sorting happens once tables are grouped not on the input tables directly. In the original query, tables are sorted and grouped during this sorting which is not very efficient if the tables are very large. With the new query, tables are grouped first and then sorted, making sorting much more efficient because grouped tables are much smaller than normal table.