GROUP ASSIGNMENT

FIT3003 ASSIGNMENT 1

Done by:

Hussain Abuwala
[27502333]

Tasin Aryan
[27477436]

10/15/2017

DETAILS OF ORACLE ACCOUNT

Account 1:

Username: S27502333
Password: student

Account 2:

Username: S27477436
Password: student

CONTRIBUTION DECLARATION FORM

Percentage of contribution:

1. Name: Hussain, ID: 27502333, Contribution: 50%

2. Name: Tasin, ID: 27477436, Contribution: 50%

List of parts that each student did:

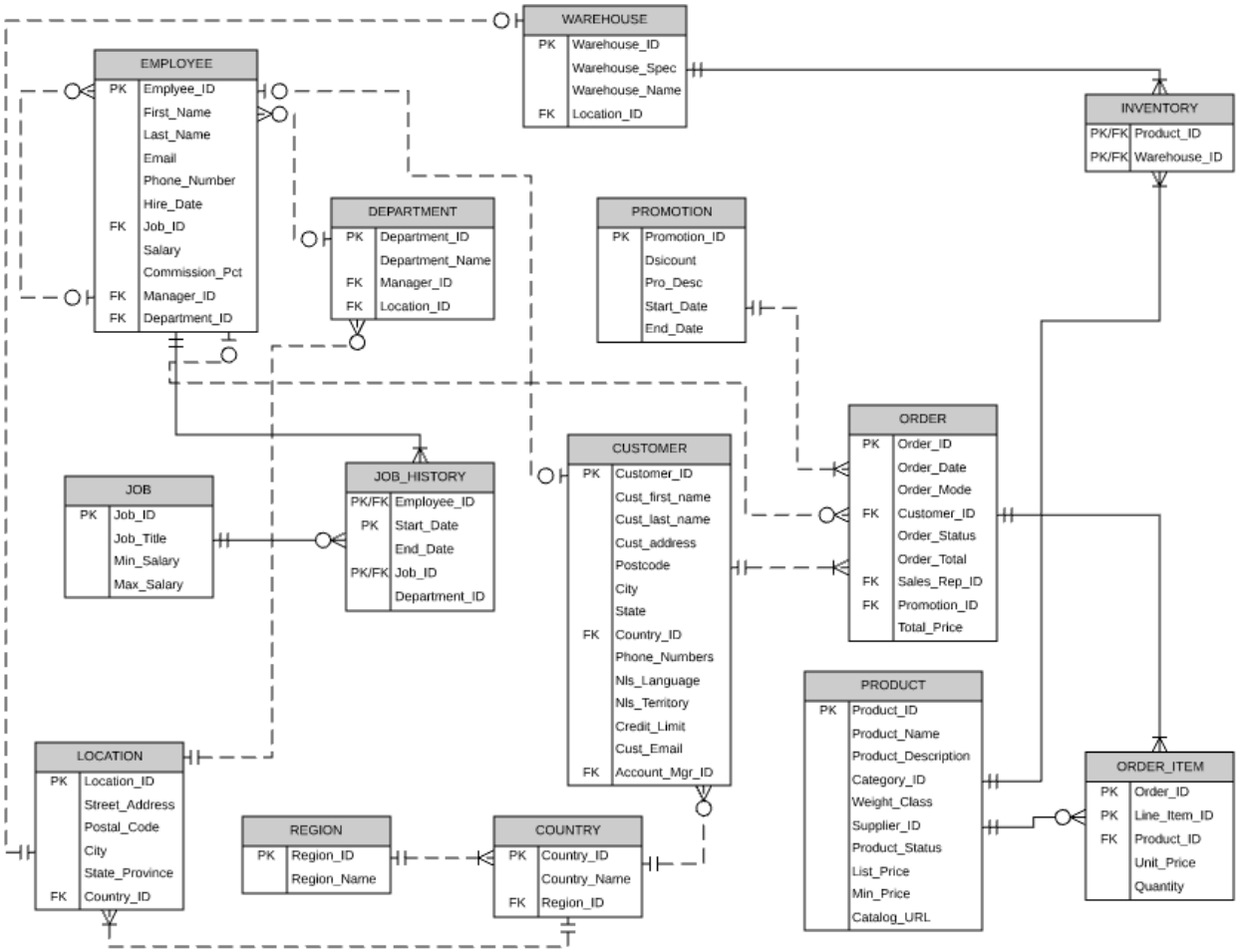
Each part was done equally

TASK C.1

(a)

Below is the link to the ER diagram that has been created using Lucid charts:
https://www.lucidchart.com/documents/view/27ca430d-d045-4513-9dee-035deec5ee05

A screenshot of the ER diagram:



P.S. Do view the link provided above for the best quality.

TASK C.1

(b)

Wrong data that we have encountered,
whether it affects our data warehouse or not and how to clean it.

1. **Same customer ID repeated twice in customers table**

Affects when trying to calculate total customers.
Need to use *distinct* when copying CUSTOMERS table.
SQL code included in TaskC.1.txt

Before:

	CUSTOMER_ID	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_ADDRESS	POSTCODE	CITY	STATE	COUNTRY_ID	PHONE_NUMBERS	NLS
1	101	Constantin	Welles	514 W Superior St	46901	Kokomo	IN	US	+1 317 123 4104	us
2	101	Constantin	Welles	514 W Superior St	46901	Kokomo	IN	US	+1 317 123 4104	us
3	102	Harrison	Pacino	2515 Bloyd Ave	46218	Indianapolis	IN	US	+1 317 123 4111	us
4	103	Manisha	Taylor	8768 N State Rd 37	47404	Bloomington	IN	US	+1 812 123 4115	us
5	104	Harrison	Sutherland	6445 Bay Harbor Ln	46254	Indianapolis	IN	US	+1 317 123 4126	us
6	105	Matthias	MacGraw	4019 W 3Rd St	47404	Bloomington	IN	US	+1 812 123 4129	us
7	106	Matthias	Hannah	1608 Portage Ave	46616	South Bend	IN	US	+1 219 123 4136	us
8	107	Matthias	Cruise	23943 Us Highway 33	46517	Elkhart	IN	US	+1 219 123 4138	us

After:

	CUSTOMER_ID	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_ADDRESS	POSTCODE	CITY	STATE	COUNTRY_ID	PHONE_NUMBERS	NLS_L
1	101	Constantin	Welles	514 W Superior St	46901	Kokomo	IN	US	+1 317 123 4104	us
2	102	Harrison	Pacino	2515 Bloyd Ave	46218	Indianapolis	IN	US	+1 317 123 4111	us
3	103	Manisha	Taylor	8768 N State Rd 37	47404	Bloomington	IN	US	+1 812 123 4115	us
4	104	Harrison	Sutherland	6445 Bay Harbor Ln	46254	Indianapolis	IN	US	+1 317 123 4126	us
5	105	Matthias	MacGraw	4019 W 3Rd St	47404	Bloomington	IN	US	+1 812 123 4129	us
6	106	Matthias	Hannah	1608 Portage Ave	46616	South Bend	IN	US	+1 219 123 4136	us
7	107	Matthias	Cruise	23943 Us Highway 33	46517	Elkhart	IN	US	+1 219 123 4138	us

2. **Same product ID with same warehouse ID repeated in inventories table**

Affects the total sales by each product and warehouse.
Need to use *distinct* when copying CUSTOMERS table.
SQL code included in TaskC.1.txt

Before:

	PRODUCT_ID	WAREHOUSE_ID
784	3083	9
785	3083	6
786	3083	4
787	3083	2
788	3086	8
789	3086	9
790	3086	6
791	3086	2
792	3086	4
793	3086	4
794	3088	8
795	3088	9
796	3088	6
797	3088	4

	PRODUCT_ID	WAREHOUSE_ID
550	2418	4
551	2418	1
552	2419	9
553	2419	2
554	2419	6
555	2419	7
556	2419	5
557	2419	1
558	2419	2
559	2419	3
560	2419	4
561	2419	8
562	2422	6
563	2422	7

After:

	PRODUCT_ID	WAREHOUSE_ID
781	3083	6
782	3083	2
783	3083	9
784	3083	8
785	3086	4
786	3086	8
787	3086	6
788	3086	9
789	3086	2
790	3088	4
791	3088	9

	PRODUCT_ID	WAREHOUSE_ID
549	2418	4
550	2419	6
551	2419	5
552	2419	9
553	2419	1
554	2419	4
555	2419	2
556	2419	8
557	2419	3
558	2419	7
559	2422	7

3. **Same order ID with different attributes in <u>orders</u> table**

Does not affect. No need to clean.

4. **Min salary greater than max salary in <u>employees</u> table**

Does not affect. No need to clean.

5. **Salary of an employee is negative in employees table**

Affects when calculating total salary.
Need to update employees table and make the value of salary positive. Here I am assuming that the salary was mistakenly entered as negative rather than positive. Hence I will change that value to positive.
SQL code included in TaskC.1.txt

Before:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	(null)	103	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	(null)	103	60
6	105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	(null)	103	60
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	-4800	(null)	103	60
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	(null)	103	60
9	108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12008	(null)	108	100
10	109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	(null)	108	100
11	110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	(null)	108	100

After:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	(null)	103	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	(null)	103	60
6	105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	(null)	103	60
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	4800	(null)	103	60
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	(null)	103	60
9	108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12008	(null)	108	100
10	109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000	(null)	108	100
11	110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200	(null)	108	100

6. **Manager ID is not in the employee table of the company in employees table**

Does not affect. No need to clean.

7. **List price less than minimum price in products table**

Does not affect. No need to clean.

8. **Quantity is 0 in some of the order items table**

Affects when average sales need to be calculated.

Need to delete rows in order_items table which have quantity 0.
SQL code included in TaskC.1.txt

Before:

	ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	UNIT_PRICE	QUANTITY
614	2447	11	2308	56	40
615	2447	12	2311	93	44
616	2448	1	3106	48	3
617	2448	2	3114	101	0
618	2448	5	3133	43	11
619	2448	6	3134	17	14

	ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	UNIT_PRICE	QUANTITY
640	2453	1	2492	43	3
641	2454	1	2289	48	120
642	2454	2	2293	99	0
643	2454	3	2299	76	3
644	2454	6	2308	56	12

After:

	ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	UNIT_PRICE	QUANTITY
615	2447	9	2299	76	35
616	2448	6	3134	17	14
617	2448	7	3139	21	15
618	2448	1	3106	48	3
619	2448	5	3133	43	11
620	2448	8	3143	16	16
621	2449	1	2522	40	2

	ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	UNIT_PRICE	QUANTITY
639	2453	1	2492	43	3
640	2454	6	2308	56	12
641	2454	7	2316	21	13
642	2454	1	2289	48	120
643	2454	9	2334	3.3	18
644	2454	3	2299	76	3
645	2454	8	2323	18	16
646	2455	12	2536	80	54

TASK C.2

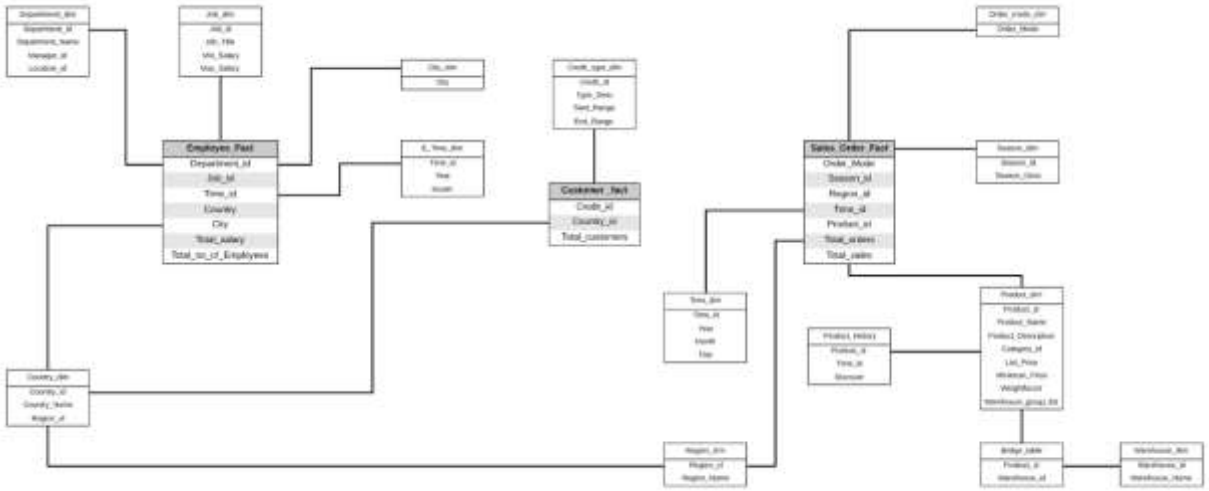
(a)

Two queries

- What are the top 3 selling products by each year?
- What are the top 2 most employed jobs by each country?

(b)

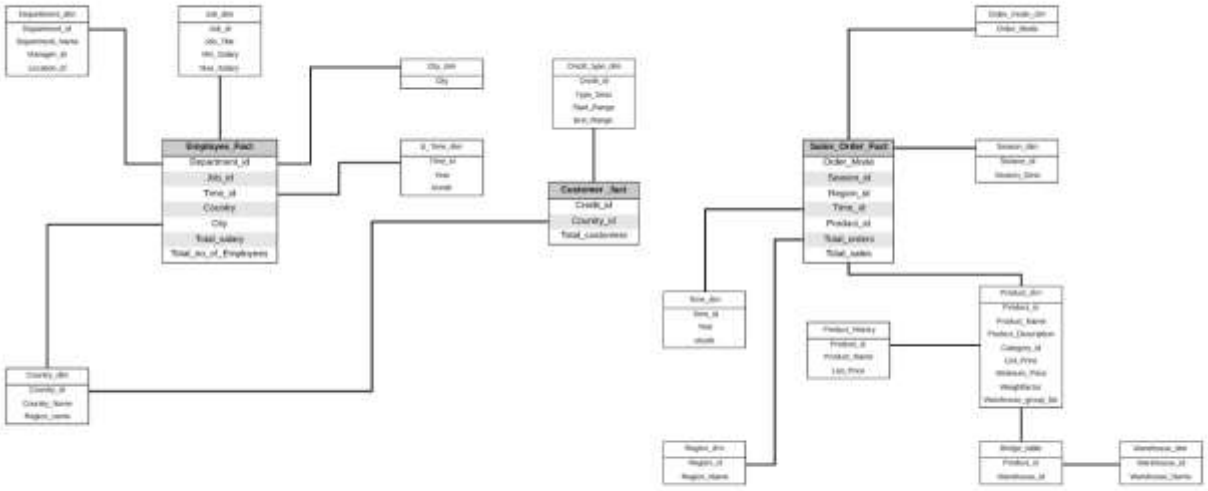
Version-1



High resolution document:

https://www.lucidchart.com/documents/view/39a68030-1331-43ce-add3-a00f698e4367

Version-2



High resolution document:

https://www.lucidchart.com/documents/view/50970e67-b433-45c8-a500-03a434411eb9

(c)

	Version-1	Version-2
Hierarchy	√ **(**Employee_Fact)	None
Bridge Table	√ (Sales_Order_Fact)	√ (Sales_Order_Fact)
Temporal	SCD TYPE-4 (Sales_Order_Fact)	SCD TYPE-0 (Sales_Order_Fact)

Note: "√" above means that Version-1 or Version-2 contains a specific feature,
"None" above means that a version does not contain a specific feature.

A short explanation on the star schemas

- In version-1 we used hierarchy of "Country" to "Region" in the Employee_Fact on the other hand version-2 has no hierarchy.

- In version-1 and version-2 we used the same bridge table to connect the "Product dimension" and "Warehouse dimension" tables.

- In version-1 we used SCD TYPE-4 temporal technique which keeps all the history of the product prices over time. In version-2 we used SCD TYPE-0 temporal technique which only keeps the initial price **(ListPrice)** of any product entered in the database.

- In the First star schema, as we have SCD TYPE-4 we needed the time dimension to have year, month and day as price of that product can change any day. But in star schema version 2, as we are using SCD TYPE-0, the price of the product stays the same always so my time dimension is more aggregated leaving out day and only having year and month.

TASK C.3

(a)

SQL commands for Star Schema V1 included in TaskC.3.txt

(b)

SQL commands for Star Schema V1 included in TaskC.3.txt

(c)

Screenshots of the tables created.

- product_history

	PRODUCT_ID	TIME_ID	DISCOUNT
1	1781	20080108	2.75
2	1782	20080108	0
3	1787	20060227	0
4	1791	20060227	4.4
5	1797	20070831	9.23
6	1797	20080108	9.23
7	1797	20060227	9.23
8	1799	20060227	3.86
9	1803	20080108	8.33
10	1806	20070831	10

- order_mode_DIM

	ORDER_MODE
1	direct
2	online

- region_DIM

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

- season_DIM

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

- bridge_table

	PRODUCT_ID	WAREHOUSE_ID
1	1733	1
2	2377	1
3	2380	1
4	2418	1
5	3000	1
6	3139	1
7	3300	1
8	1742	2
9	1750	2
10	1769	2

- WarehouseToBridge

	WAREHOUSE_ID	WAREHOUSE_NAME
1	1	Southlake, Texas
2	2	San Francisco
3	3	New Jersey
4	4	Seattle, Washington
5	5	Toronto
6	6	Sydney
7	7	Mexico City
8	8	Beijing
9	9	Bombay

- product_DIM

	PRODUCT_ID	PRODUCT_NAME	PRODUCT_DESCRIPTION	CATEGORY_ID	LIST_PRICE	MIN_PRICE	WEIGHTFACTOR	WAREHOUSEGROUPLIST
1	3000	Laptop 32/10/56	Envoy Laptop, 32MB memory, ...	19	1749	1542	0.1111111111...	1_2_3_4_5_6_7_8_9
2	3300	Screws <3.32.P>	Screws: Steel, size 32mm, P...	19	23	19	0.1111111111...	1_2_3_4_5_6_7_8_9
3	3400	HD 8GB /SE	8GB capacity SCSI hard disk...	13	328	337	0.2	2_4_6_8_9
4	1729	Chemicals - RCP	Cleaning Chemicals - 3500 r...	39	80	66	0.1666666666...	3_5_6_7_8_9
5	1733	PS 220V /UK	220V Power supply type - Un...	19	89	76	0.1111111111...	1_2_3_4_5_6_7_8_9
6	1734	Cable RS232 10/AM	10 ft RS232 cable with M/M ...	19	6	5	0.1111111111...	1_2_3_4_5_6_7_8_9
7	1737	Cable SCSI 10/FW/ADS	10ft SCSI2 F/W Adapt to DSx...	19	8	6	0.1111111111...	1_2_3_4_5_6_7_8_9
8	1738	PS 110V /US	110 V Power Supply - US com...	19	86	70	0.1111111111...	1_2_3_4_5_6_7_8_9
9	1739	SDRAM - 128 MB	SDRAM memory, 128 MB capaci...	14	299	248	0.2	2_4_6_8_9
10	1740	TD 12GB/DAI	Tape drive - 12 gigabyte ca...	17	134	111	0.2	2_4_6_8_9

- time_DIM

	TIME_ID	YEAR	MONTH	DAY
1	20071002	2007	10	02
2	20080108	2008	01	08
3	20060126	2006	01	26
4	20070828	2007	08	28
5	20080226	2008	02	26
6	20070607	2007	06	07
7	20070311	2007	03	11
8	20070701	2007	07	01
9	20070312	2007	03	12
10	20060901	2006	09	01

- sales_temp_fact_v1

	ORDER_ID	ORDER_MODE	ORDER_DATE	REGION_ID	TIME_ID	PRODUCT_ID	TOTAL_SALES	SEASON_ID
1	2355	online	26-JAN-06 07.22.51.962632000 AM	2	20060126	2289	9600	1
2	2356	online	26-JAN-08 07.22.41.934562000 AM	2	20080126	2264	7565.8	1
3	2357	direct	08-JAN-06 06.19.44.123456000 PM	2	20060108	2211	462	1
4	2358	direct	08-JAN-08 03.03.12.654278000 PM	2	20080108	1781	2039.4	1
5	2359	online	08-JAN-06 07.34.13.112233000 PM	2	20060108	2337	270.6	1
6	2359	direct	01-JAN-08 03.03.12.654278000 PM	2	20080101	2337	270.6	1
7	2360	online	14-NOV-07 10.22.31.223344000 AM	2	20071114	2058	667	4
8	2361	online	13-NOV-07 11.34.21.986210000 AM	2	20071113	2289	8640	4
9	2362	online	13-NOV-07 12.41.10.619477000 PM	2	20071113	2289	9600	4
10	2363	online	23-OCT-07 02.49.56.346122000 PM	2	20071023	2264	1791.9	3

- sales_fact_V1

251	online	1	2	20070226	3106	7008	2
252	direct	2	2	20070721	3106	3024	1
253	direct	2	2	20070709	2976	208	1
254	direct	4	2	20061116	3193	13.2	1
255	direct	3	2	20071002	3350	8316	2
256	direct	3	2	20070920	2471	1448.7	1
257	direct	3	2	20070816	3117	10640	2

- credit_type_DIM

	CREDIT_ID	TYPE_DESC	START_RANGE	END_RANGE
1	1	Low	0	1500
2	2	Medium	1501	3500
3	3	High	3501	6000

- country_DIM

	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

- Customer_temp_fact_v1

	CUSTOMER_ID	COUNTRY_ID	CREDIT_LIMIT	CREDIT_ID
1	241	US	2400	2
2	244	US	2400	2
3	246	US	2400	2
4	247	US	2400	2
5	248	US	2400	2
6	264	US	3600	3
7	309	CN	1200	1
8	126	US	300	1
9	133	US	400	1
10	134	US	400	1

- e_time_DIM

	TIME_ID	YEAR	MONTH
1	200101	2001	01
2	200206	2002	06
3	200208	2002	08
4	200212	2002	12
5	200305	2003	05
6	200306	2003	06
7	200307	2003	07
8	200309	2003	09
9	200310	2003	10
10	200401	2004	01

- city_DIM

	CITY
1	Venice
2	Tokyo
3	Sydney
4	London
5	Whitehorse
6	Bombay
7	Stretford
8	Seattle
9	Mexico City
10	Beijing

- job_DIM

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	AC_MGR	Accounting Manager	8200	16000
2	AC_ACCOUNT	Public Accountant	4200	9000
3	SH_CLERK	Shipping Clerk	2500	5500
4	PR_REP	Public Relations Representative	4500	10500
5	AD_PRES	President	20080	40000
6	FI_ACCOUNT	Accountant	4200	9000
7	PU_CLERK	Purchasing Clerk	2500	5500
8	ST_MAN	Stock Manager	5500	4500
9	IT_PROG	Programmer	4000	10000
10	FI_MGR	Finance Manager	8200	16000

- department_DIM

	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

- location_DIM

	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1	1000	1297 Via Cola di Rie	00989	Roma	(null)	IT
2	1100	93091 Calle della Testa	10934	Venice	(null)	IT
3	1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefecture	JP
4	1300	9450 Kamiya-cho	6823	Hiroshima	(null)	JP
5	1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
6	1500	2011 Interiors Blvd	99236	South San Francisco	California	US
7	1600	2007 Zagora St	50090	South Brunswick	New Jersey	US
8	1700	2004 Charade Rd	98199	Seattle	Washington	US
9	1800	147 Spadina Ave	M5V 2L7	Toronto	Ontario	CA
10	1900	6092 Boxwood St	YSW 9T2	Whitehorse	Yukon	CA

- Employee_fact_V1

	DEPARTMENT_ID	JOB_ID	TIME_ID	COUNTRY_ID	CITY	TOTAL_EMPLOYEES	TOTAL_SALARY
1	90	AD_VP	200101	US	Seattle	1	17000
2	110	AC_ACCOUNT	200206	US	Seattle	1	9300
3	110	AC_MGR	200206	US	Seattle	1	12008
4	40	HR_REP	200206	UK	London	1	6500
5	70	PR_REP	200206	DE	Munich	1	10000
6	100	FI_ACCOUNT	200208	US	Seattle	1	9000
7	100	FI_MGR	200208	US	Seattle	1	12008
8	30	PU_MAN	200212	US	Seattle	1	11000
9	30	PU_CLERK	200305	US	Seattle	1	3100
10	50	ST_MAN	200305	US	South San Francisco	1	7900

All the screenshots shown above are of Star schema version 1. For star schema version 2 most of the tables are same. The ones that are different are shown below.

- product_history_V2

	PRODUCT_ID	PRODUCT_NAME	ORIGINAL_PRICE
1	1726	LCD Monitor 11/PM	259
2	2359	LCD Monitor 9/PM	249
3	3060	Monitor 17/HR	299
4	2243	Monitor 17/HR/F	350
5	3057	Monitor 17/SD	369
6	3061	Monitor 19/SD	499
7	2245	Monitor 19/SD/M	512
8	3065	Monitor 21/D	999
9	3331	Monitor 21/HR	879

(*note that the original_price dimension above is named ListPrice in the original SQL.)

- country_DIM_V2

	COUNTRY_ID	COUNTRY_NAME	REGION_NAME
1	AR	Argentina	Americas
2	AU	Australia	Asia
3	BE	Belgium	Europe
4	BR	Brazil	Americas
5	CA	Canada	Americas
6	CH	Switzerland	Europe
7	CN	China	Asia
8	DE	Germany	Europe
9	DK	Denmark	Europe

- time_DIM_V2

	TIME_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

(d)

A comparison of the star schemas

- In version-1 we used hierarchy of “Country” to “Region” in the Employee_Fact on the other hand version-2 has no hierarchy.

- In version-1 and version-2 we used the same bridge table to connect the “Product dimension” and “Warehouse dimension” tables.

- In version-1 we used SCD TYPE-4 temporal technique which keeps all the history of the product prices over time. In version-2 we used SCD TYPE-0 temporal technique which only keeps the initial price **(ListPrice)** of any product entered in the database.

- In the First star schema, as we have SCD TYPE-4 we needed the time dimension to have year, month and day as price of that product can change any day. But in star schema version 2, as we are using SCD TYPE-0, the price of the product stays the same always so my time dimension is more aggregated leaving out day and only having year and month.

From the query point of view for hierarchy
Both hierarchy and non-hierarchy will need the same number of queries to answer a specific question.

From the query processing point of view hierarchy
The hierarchy version need to do join to get the result which may be expensive compared to the non-hierarchy version which does not need any join.

From the business point of view for temporal

SCD-TYPE 4 may be helpful if the company drastically and regularly changes the price of the products and is keen to see what the previous prices of its sold products were to get a better insight into a particular product sale over time.
SCD-TYPE 0 is not actually of any help as only keeping the original price is of no use to find a better insight into the sales of a company unless the company is such whose products are always of a fixed price and never change which is obviously very rare.

TASK C.4

N.B. ALL THE SQL COMMANDS FOR THE BELOW QUERIES HAVE BEEN INCLUDED IN TASKC.4.TXT

(a) Task C.4 Reports with Proper sub-totals

Queries

--What is the total salary by each department and each job in each month?

This is useful because the total salary taken from these criteria can be used by the company to keep track of money spent on salary for efficient management.

Query result using rollup

	DEPARTMENTNAME	JOBTITLE	MONTH	TOTAL_SALARY
80	Purchasing	Purchasing Manager	All_months	11000
81	Purchasing	All_jobs	All_months	24900
82	Administration	Administration As...	09	4400
83	Administration	Administration As...	All_months	4400
84	Administration	All_jobs	All_months	4400
85	Human Resources	Human Resources R...	06	6500
86	Human Resources	Human Resources R...	All_months	6500
87	Human Resources	All_jobs	All_months	6500
88	Public Relations	Public Relations ...	06	10000
89	Public Relations	Public Relations ...	All_months	10000
90	Public Relations	All_jobs	All_months	10000
91	All_Departments	All_jobs	All_months	685416

Query result using cube

	DEPARTMENTNAME	JOBTITLE	MONTH	TOTAL_SALARY
1	All_Departments	All_jobs	All_months	685416
2	All_Departments	All_jobs	01	108500
3	All_Departments	All_jobs	02	60600
4	All_Departments	All_jobs	03	128400
5	All_Departments	All_jobs	04	36600
6	All_Departments	All_jobs	05	31000
7	All_Departments	All_jobs	06	82208
8	All_Departments	All_jobs	07	26200
9	All_Departments	All_jobs	08	57308
10	All_Departments	All_jobs	09	40000
11	All_Departments	All_jobs	10	41000
12	All_Departments	All_jobs	11	32900
13	All_Departments	All_jobs	12	40700
14	All_Departments	President	All_months	24000
15	All_Departments	President	06	24000

--What is the number of employees by each country and by each city?

This is useful to keep track of employee numbers and to calculate proportion of employees from each country and city.

This data can be used to visualize the diversity of employees in case it is needed to be presented.

Query result using rollup

	COUNTRY_NAME	CITY	TOTAL_EMP
1	Canada	Toronto	2
2	Canada	All_City	2
3	Germany	Munich	1
4	Germany	All_City	1
5	United Kingdom	London	1
6	United Kingdom	Oxford	34
7	United Kingdom	All_City	35
8	United States of America	Seattle	18
9	United States of America	Southlake	5
10	United States of America	South San Francisco	45
11	United States of America	All_City	68
12	All_Country	All_City	106

Query result using cube

	COUNTRY_NAME	CITY	TOTAL_EMP
1	All_Country	All_City	106
2	All_Country	London	1
3	All_Country	Munich	1
4	All_Country	Oxford	34
5	All_Country	Seattle	18
6	All_Country	Toronto	2
7	All_Country	Southlake	5
8	All_Country	South San Francisco	45
9	Canada	All_City	2
10	Canada	Toronto	2
11	Germany	All_City	1
12	Germany	Munich	1
13	United Kingdom	All_City	35
14	United Kingdom	London	1
15	United Kingdom	Oxford	34
16	United States of America	All_City	68
17	United States of America	Seattle	18
18	United States of America	Southlake	5
19	United States of America	South San Francisco	45

(b) Reports with Rank and Percent_Rank

Queries

--What are the top 3 selling products?

This is useful as this result can be used to know the top selling products of the company.

Query result

	PRODUCT_NAME	TOTAL_ORDERS	ORDERS_RANK
1	Screws <B.28.S>	18	1
2	LaserPro 600/6/BW	17	2
3	Mouse C/E	15	3

--What is the top 10% revenue?.

This is useful as we can know the years which had the highest revenue where we would only be interested in the top 10% of those years.

Query result

	YEAR	MONTH	TOTAL_REVENUE	Percent Rank
1	2007	06	352186.8	0.916666666666666666666666666666666666666667
2	2007	09	367945.9	0.958333333333333333333333333333333333333333
3	2007	11	398487.1	1

(c) Reports with Partitions

Queries

--What are the top 3 products by each year?

This insight is useful for knowing what the popular products were as years went by and maybe find out which products have remained at the top consistently throughout the years.

Query result

	YEAR	PRODUCT_NAME	TOTAL_ORDERS	ORDERS_RANK
58	2006	Project Management - S3.3	1	2
59	2006	Sound Card STD	1	2
60	2006	Business Cards - 1000/2L	1	2
61	2007	LaserPro 600/6/BW	12	1
62	2007	Screws <B.28.S>	11	2
63	2007	Mouse C/E	9	3
64	2008	Sound Card STD	6	1
65	2008	Screws <B.28.S>	5	2
66	2008	Mouse C/E	5	2
67	2008	LaserPro 600/6/BW	3	3
68	2008	Screws <S.16.S>	3	3
69	2008	Manual - Vision Tools2.0	3	3
70	2008	Manual - Vision OS/2.x	3	3

--What are the top 2 most employed jobs by each country?

This gives us insight about the most hiring jobs in each country. This data can be used to know what jobs to promote in each countries etc.

Query result

	COUNTRY_NAME	JOB_TITLE	TOTAL_EMPLOYEES	RANK
1	Canada	Marketing Manager	1	1
2	Canada	Marketing Representative	1	1
3	Germany	Public Relations Representative	1	1
4	United Kingdom	Sales Representative	29	1
5	United Kingdom	Sales Manager	5	2
6	United States of America	Shipping Clerk	20	1
7	United States of America	Stock Clerk	20	1
8	United States of America	Accountant	5	2
9	United States of America	Purchasing Clerk	5	2
10	United States of America	Programmer	5	2
11	United States of America	Stock Manager	5	2

(d) Reports with cumulative and moving aggregate

Queries

--What are the total cumulative sales by each quarter of each year?

This query result makes it easier to know the total amount of sales up until the quarter of a year, which makes calculation faster and minimizes error as repeated calculation is not required.

Query result

	SEASON_ID	SEASON_DESC	YEAR	TOTAL_SALES	CUM_SALES
1	1	Spring	2004	5,606	5,606
2	1	Spring	2006	385,702	391,308
3	2	Summer	2006	63,508	454,816
4	3	Autumn	2006	19,119	473,935
5	4	Winter	2006	12,543	486,478
6	1	Spring	2007	318,637	805,115
7	2	Summer	2007	801,119	1,606,234
8	3	Autumn	2007	709,316	2,315,549
9	4	Winter	2007	740,585	3,056,134
10	1	Spring	2008	141,048	3,197,183
11	2	Summer	2008	512,641	3,709,824
12	3	Autumn	2008	2,075	3,711,899

--What are the total cumulative orders by each quarter of each year with moving window of 2 quarters?

This query does the same thing as the cumulative shown above but here it calculates total orders and has a modified functionality that it totals the orders by 2 quarters of a year. This is useful as sometimes we might need recent information like total orders for the last two seasons, rather than total orders since 2004 (which normal cumulative function would output).

Query result

	SEASON_ID	SEASON_DESC	YEAR	TOTAL_ORDERS	CUM_ORDERS
1	1	Spring	2004	6	6
2	1	Spring	2006	56	62
3	2	Summer	2006	23	85
4	3	Autumn	2006	12	91
5	4	Winter	2006	10	45
6	1	Spring	2007	68	90
7	2	Summer	2007	137	215
8	3	Autumn	2007	118	323
9	4	Winter	2007	103	358
10	1	Spring	2008	53	274
11	2	Summer	2008	85	241
12	3	Autumn	2008	2	140

TASK C.5

SQL commands included in TaskC.5.txt

Techniques Used:

- **/*+ ORDERED */**
- **/*+ USE_MERGE (e c) */**
- **/*+ USE_NL (s p) */**
- **/*+ no_merge */**
    - **“With grouping as” with nested query**
    - **Only nested query**
    - **Hash aggregation**

Below, each question has been answered using two queries that behave differently but produce the same output. The first query is from question 4 and the second query is from question 5.

For each of the queries I have included

- **Execution Plan**
- **Query Result**
- **Query Tree**

REPORTS WITH PROPER SUB-TOTALS

--What is the total salary by each department and each job in each month?

FIRST QUERY (Q4)

Rollup

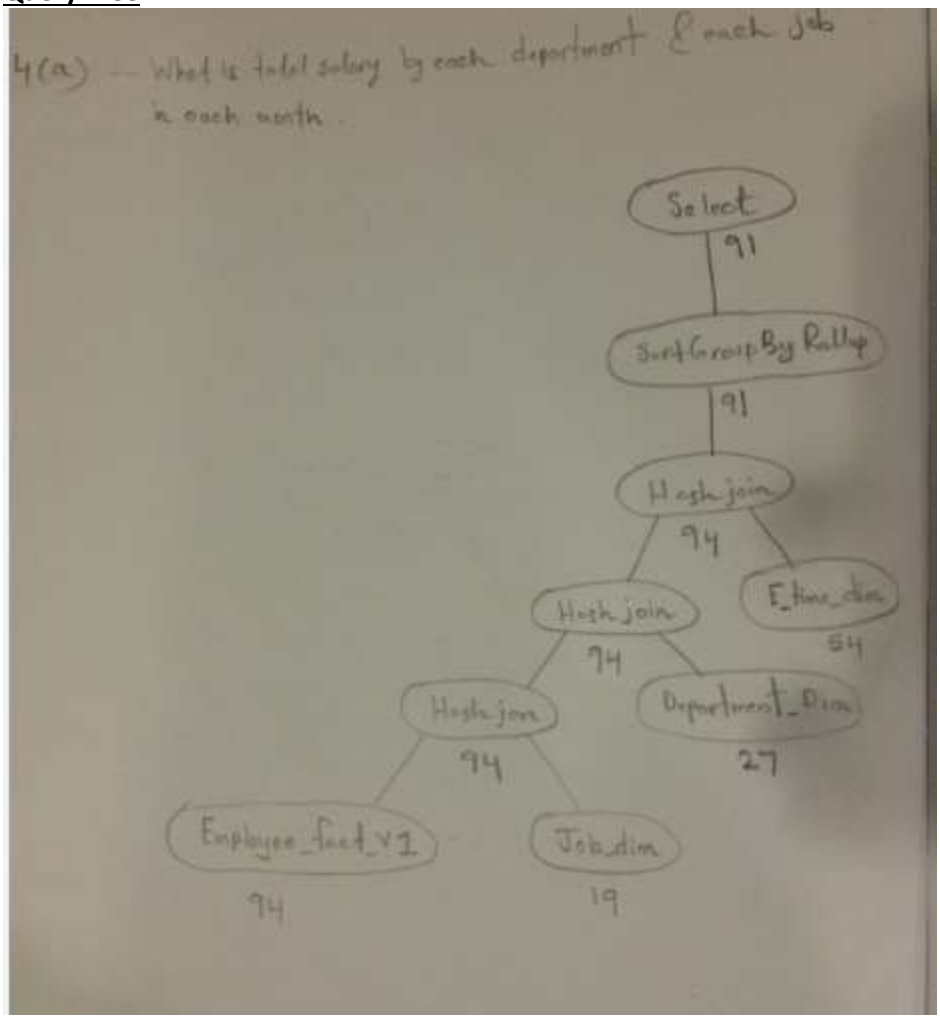
```
Plan hash value: 3516553657

-----------------------------------------------------------------------------------------------
| Id  | Operation             | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                  |    91 |  7007 |    13   (8)| 00:00:01 |
|   1 |  SORT GROUP BY ROLLUP |                  |    91 |  7007 |    13   (8)| 00:00:01 |
|*  2 |   HASH JOIN           |                  |    94 |  7238 |    12   (0)| 00:00:01 |
|*  3 |    HASH JOIN          |                  |    94 |  6298 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN         |                  |    94 |  4794 |     6   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| JOB_DIM          |    19 |   513 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| EMPLOYEE_FACT_V1 |    94 |  2256 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | DEPARTMENT_DIM   |    27 |   432 |     3   (0)| 00:00:01 |
|   8 |    TABLE ACCESS FULL  | E_TIME_DIM       |    54 |   540 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result using rollup

	DEPARTMENTNAME	JOBTITLE	MONTH	TOTAL_SALARY
80	Purchasing	Purchasing Manager	All_months	11000
81	Purchasing	All_jobs	All_months	24900
82	Administration	Administration As...	09	4400
83	Administration	Administration As...	All_months	4400
84	Administration	All_jobs	All_months	4400
85	Human Resources	Human Resources R...	06	6500
86	Human Resources	Human Resources R...	All_months	6500
87	Human Resources	All_jobs	All_months	6500
88	Public Relations	Public Relations ...	06	10000
89	Public Relations	Public Relations ...	All_months	10000
90	Public Relations	All_jobs	All_months	10000
91	All_Departments	All_jobs	All_months	685416

Query Tree


4(a) -- What is total salary by each department & each job in each month.
Select
91
SortGroupBy Rollup
91
Hash join
94
Hash join
74
E_time_dim
54
Hash join
94
Department_Dim
27
Employee_fact_V1
94
Job_dim
19

SECOND QUERY (Q5)

Rollup

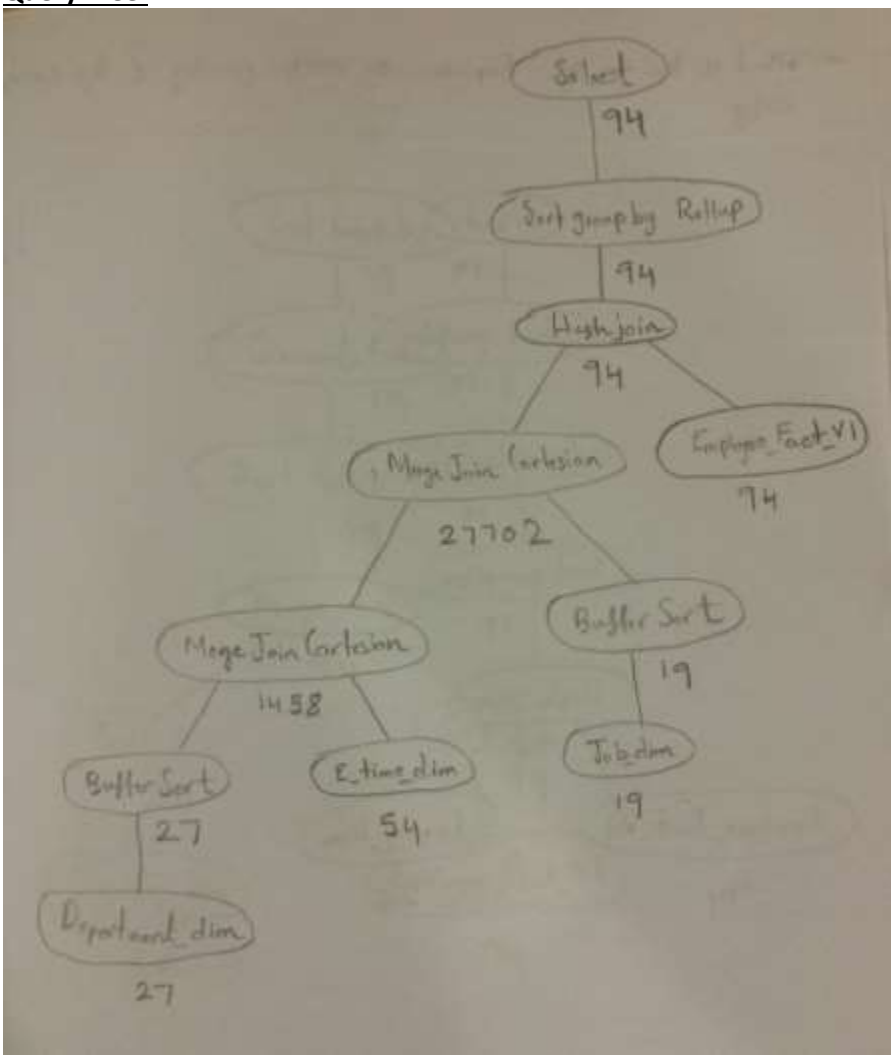
```
Plan hash value: 581363468

-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                   |    94 |  7238 |  1649   (1)| 00:00:01 |
|   1 |  SORT GROUP BY ROLLUP    |                   |    94 |  7238 |  1649   (1)| 00:00:01 |
|*  2 |   HASH JOIN              |                   |    94 |  7238 |  1648   (1)| 00:00:01 |
|   3 |    TABLE ACCESS FULL     | EMPLOYEE_FACT_V1  |    94 |  2256 |     3   (0)| 00:00:01 |
|   4 |    MERGE JOIN CARTESIAN  |                   | 27702 |  1433K|  1645   (1)| 00:00:01 |
|   5 |     MERGE JOIN CARTESIAN |                   |  1458 | 37908 |    63   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL   | E_TIME_DIM        |    54 |   540 |     3   (0)| 00:00:01 |
|   7 |      BUFFER SORT         |                   |    27 |   432 |    60   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL  | DEPARTMENT_DIM    |    27 |   432 |     1   (0)| 00:00:01 |
|   9 |     BUFFER SORT          |                   |    19 |   513 |  1644   (1)| 00:00:01 |
|  10 |      TABLE ACCESS FULL   | JOB_DIM           |    19 |   513 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result using Rollup

	DEPARTMENTNAME	JOBTITLE	MONTH	TOTAL_SALARY
80	Purchasing	Purchasing Manager	All_months	11000
81	Purchasing	All_jobs	All_months	24900
82	Administration	Administration As...	09	4400
83	Administration	Administration As...	All_months	4400
84	Administration	All_jobs	All_months	4400
85	Human Resources	Human Resources R...	06	6500
86	Human Resources	Human Resources R...	All_months	6500
87	Human Resources	All_jobs	All_months	6500
88	Public Relations	Public Relations ...	06	10000
89	Public Relations	Public Relations ...	All_months	10000
90	Public Relations	All_jobs	All_months	10000
91	All_Departments	All_jobs	All_months	685416

Query Tree:



COMPARISON: More rows are carried through the pipeline using the second query where Cartesian product is used. First query is more efficient as fewer rows are carried through each Join.

--What is the number of employees by each country and by each city?

FIRST QUERY (Q4)

CUBE

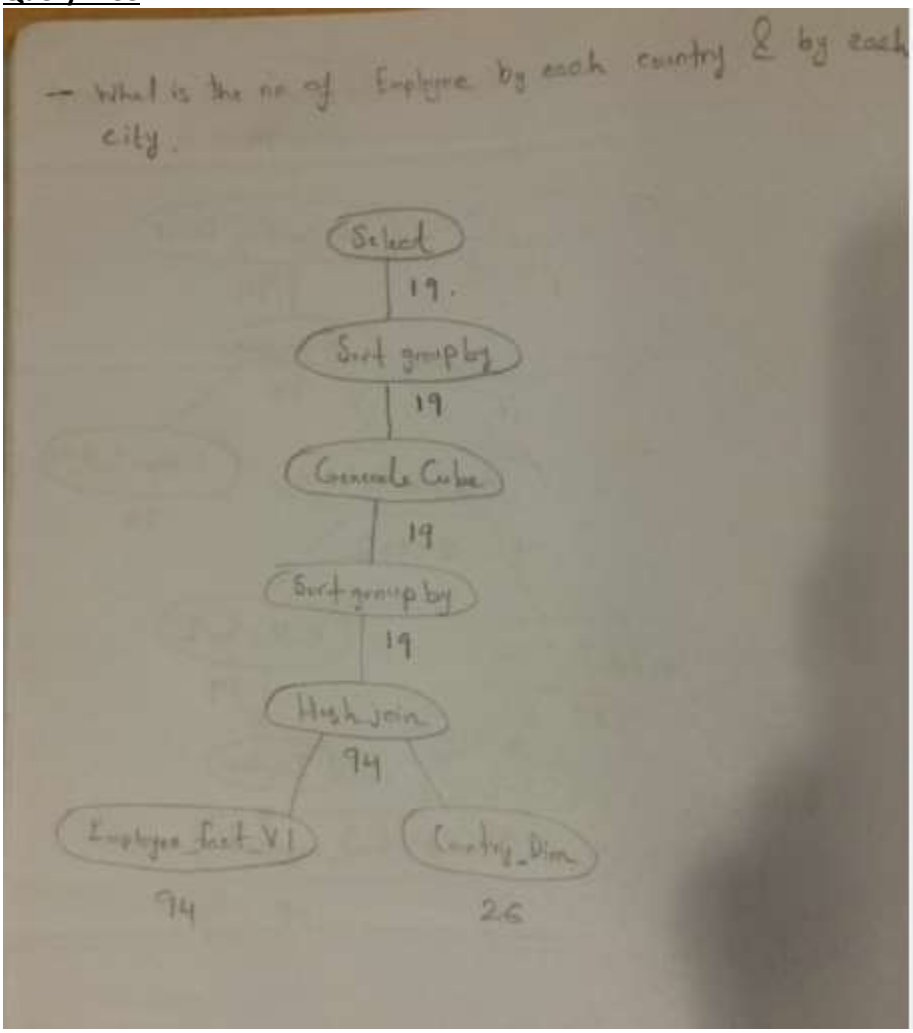
```
Plan hash value: 772264755

-----------------------------------------------------------------------------------------------------
| Id  | Operation             | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                   |    19 |   608 |     7  (15)| 00:00:01 |
|   1 |  SORT GROUP BY        |                   |    19 |   608 |     7  (15)| 00:00:01 |
|   2 |   GENERATE CUBE       |                   |    19 |   608 |     7  (15)| 00:00:01 |
|   3 |    SORT GROUP BY      |                   |    19 |   608 |     7  (15)| 00:00:01 |
|*  4 |     HASH JOIN         |                   |    94 |  3008 |     6   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| COUNTRY_DIM       |    26 |   312 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| EMPLOYEE_FACT_V1  |    94 |  1880 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
```

Query result using cube

	COUNTRY_NAME	CITY	TOTAL_EMP
1	All_Country	All_City	106
2	All_Country	London	1
3	All_Country	Munich	1
4	All_Country	Oxford	34
5	All_Country	Seattle	18
6	All_Country	Toronto	2
7	All_Country	Southlake	5
8	All_Country	South San Francisco	45
9	Canada	All_City	2
10	Canada	Toronto	2
11	Germany	All_City	1
12	Germany	Munich	1
13	United Kingdom	All_City	35
14	United Kingdom	London	1
15	United Kingdom	Oxford	34
16	United States of America	All_City	68
17	United States of America	Seattle	18
18	United States of America	Southlake	5
19	United States of America	South San Francisco	45

Query Tree


– What is the no of Employee by each country & by each city.
Select
19
Sort group by
19
Generate Cube
19
Sort group by
19
Hash join
94
Employee_Fact_V1
74
Country_Dim
26

SECOND QUERY (Q5)

CUBE

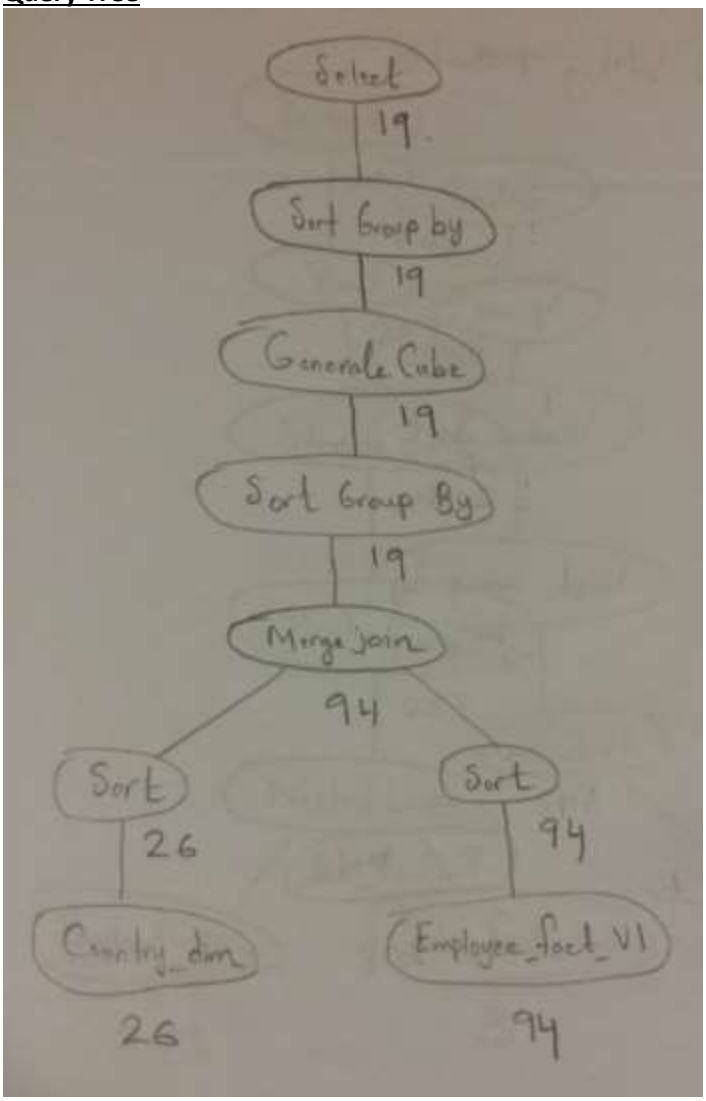
```
Plan hash value: 3529996644

-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                   |    19 |   608 |     9  (34)| 00:00:01 |
|   1 |  SORT GROUP BY         |                   |    19 |   608 |     9  (34)| 00:00:01 |
|   2 |   GENERATE CUBE        |                   |    19 |   608 |     9  (34)| 00:00:01 |
|   3 |    SORT GROUP BY       |                   |    19 |   608 |     9  (34)| 00:00:01 |
|   4 |     MERGE JOIN         |                   |    94 |  3008 |     8  (25)| 00:00:01 |
|   5 |      SORT JOIN         |                   |    26 |   312 |     4  (25)| 00:00:01 |
|   6 |       TABLE ACCESS FULL| COUNTRY_DIM       |    26 |   312 |     3   (0)| 00:00:01 |
|*  7 |      SORT JOIN         |                   |    94 |  1880 |     4  (25)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| EMPLOYEE_FACT_V1  |    94 |  1880 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Query result using cube

	COUNTRY_NAME	CITY	TOTAL_EMP
1	All_Country	All_City	106
2	All_Country	London	1
3	All_Country	Munich	1
4	All_Country	Oxford	34
5	All_Country	Seattle	18
6	All_Country	Toronto	2
7	All_Country	Southlake	5
8	All_Country	South San Francisco	45
9	Canada	All_City	2
10	Canada	Toronto	2
11	Germany	All_City	1
12	Germany	Munich	1
13	United Kingdom	All_City	35
14	United Kingdom	London	1
15	United Kingdom	Oxford	34
16	United States of America	All_City	68
17	United States of America	Seattle	18
18	United States of America	Southlake	5
19	United States of America	South San Francisco	45

Query Tree



COMPARISON: The first query uses hash join where else the second query uses sort merge which is less efficient than hash join.

REPORTS WITH RANK AND PERCENT_RANK

--Top 3 selling products

FIRST QUERY (Q4)

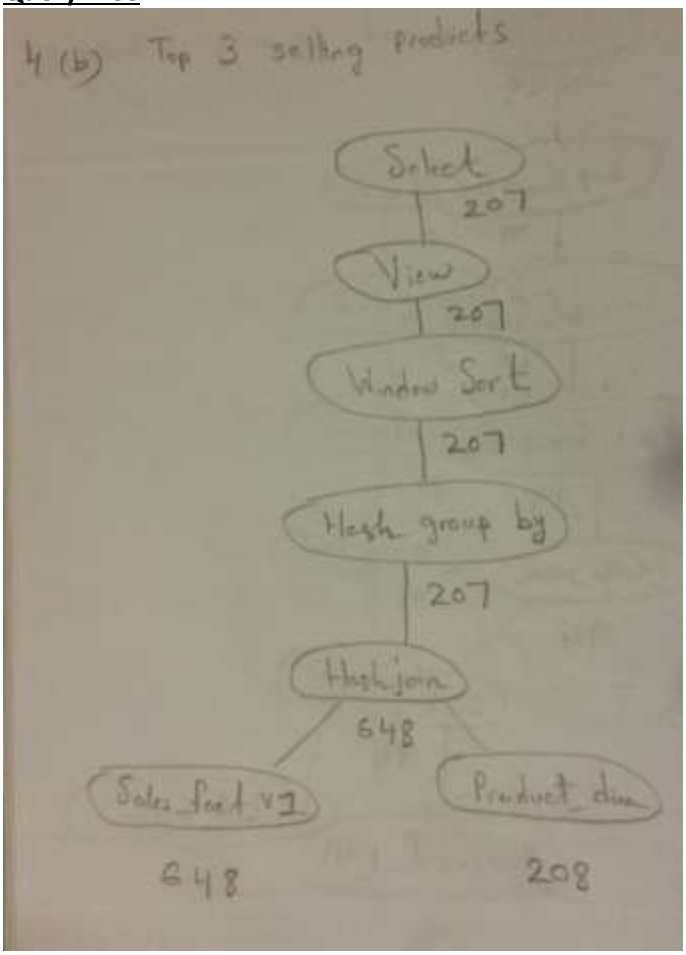
```
Plan hash value: 3200209090

-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |               |   207 | 10971 |    10  (20)| 00:00:01 |
|*  1 |  VIEW                    |               |   207 | 10971 |    10  (20)| 00:00:01 |
|*  2 |   WINDOW SORT PUSHED RANK|               |   207 |  5796 |    10  (20)| 00:00:01 |
|   3 |    HASH GROUP BY         |               |   207 |  5796 |    10  (20)| 00:00:01 |
|*  4 |     HASH JOIN            |               |   648 | 18144 |     8   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL   | PRODUCT_DIM   |   208 |  4368 |     4   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL   | SALES_FACT_V1 |   648 |  4536 |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result

	PRODUCT_NAME	TOTAL_ORDERS	ORDERS_RANK
1	Screws <B.28.S>	18	1
2	LaserPro 600/6/BW	17	2
3	Mouse C/E	15	3

Query Tree


4 (b) Top 3 selling products
Select
207
View
207
Window Sort
207
Hash group by
207
Hash join
648
Sales_fact_v1
648
Product_dim
208

SECOND QUERY (Q5)

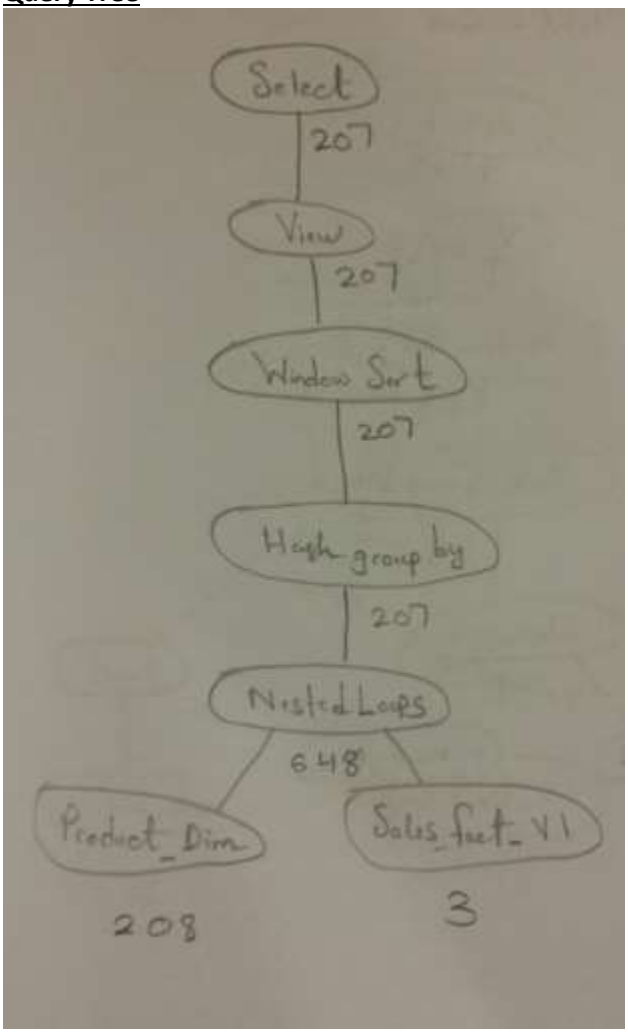
```
Plan hash value: 1179579404

-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |               |   207 | 10971 |   404   (1)| 00:00:01 |
|*  1 |  VIEW                    |               |   207 | 10971 |   404   (1)| 00:00:01 |
|*  2 |   WINDOW SORT PUSHED RANK|               |   207 |  5796 |   404   (1)| 00:00:01 |
|   3 |    HASH GROUP BY         |               |   207 |  5796 |   404   (1)| 00:00:01 |
|   4 |     NESTED LOOPS         |               |   648 | 18144 |   402   (1)| 00:00:01 |
|   5 |      TABLE ACCESS FULL   | PRODUCT_DIM   |   208 |  4368 |     4   (0)| 00:00:01 |
|*  6 |      TABLE ACCESS FULL   | SALES_FACT_V1 |     3 |    21 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result

	PRODUCT_NAME	TOTAL_ORDERS	ORDERS_RANK
1	Screws <B.28.S>	18	1
2	LaserPro 600/6/BW	17	2
3	Mouse C/E	15	3

Query Tree



COMPARISON: The first query uses hash join where else the second query uses nested loops which is less efficient than hash join.

--Top 10% total revenue

FIRST QUERY (Q4)

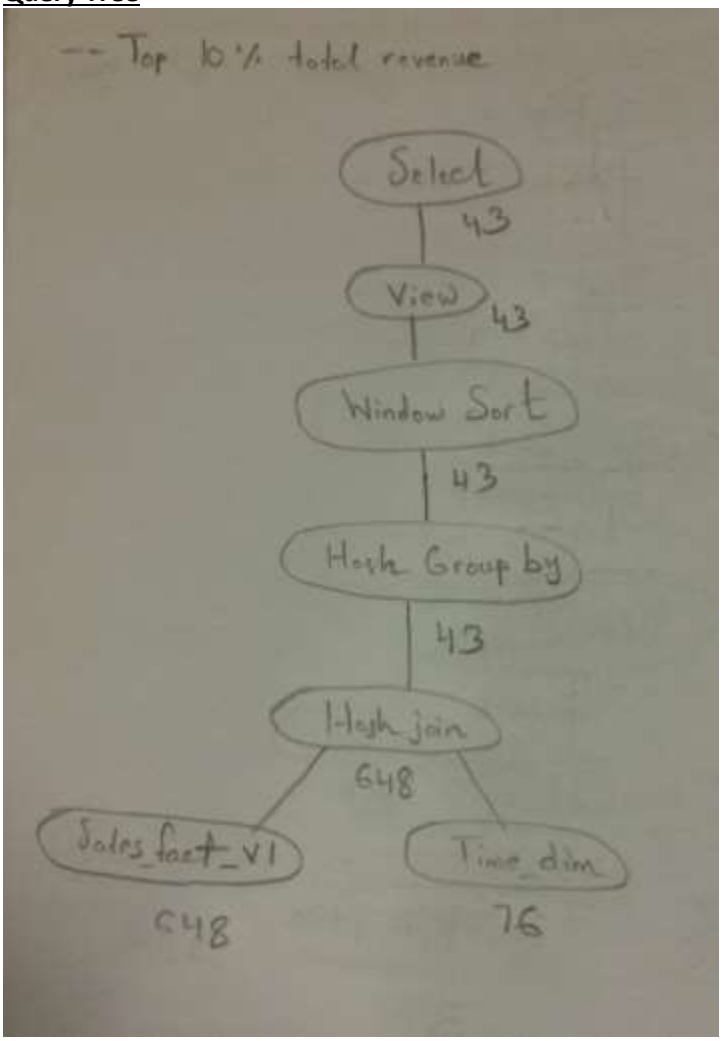
```
Plan hash value: 4079276785

-----------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               |    43 |  1419 |     9  (23)| 00:00:01 |
|*  1 |  VIEW                 |               |    43 |  1419 |     9  (23)| 00:00:01 |
|   2 |   WINDOW SORT         |               |    43 |  1333 |     9  (23)| 00:00:01 |
|   3 |    HASH GROUP BY      |               |    43 |  1333 |     9  (23)| 00:00:01 |
|*  4 |     HASH JOIN         |               |   648 | 20088 |     7   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| TIME_DIM      |    76 |  1292 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| SALES_FACT_V1 |   648 |  9072 |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result

	YEAR	MONTH	TOTAL_REVENUE	Percent Rank
1	2007	06	352186.8	0.916666666666666666666666666666666666666667
2	2007	09	367945.9	0.958333333333333333333333333333333333333333
3	2007	11	398487.1	1

Query Tree


-- Top 10% total revenue
Select
43
View
43
Window Sort
43
Hash Group by
43
Hash join
648
Sales_fact_V1
648
Time_dim
76

SECOND QUERY (Q5)

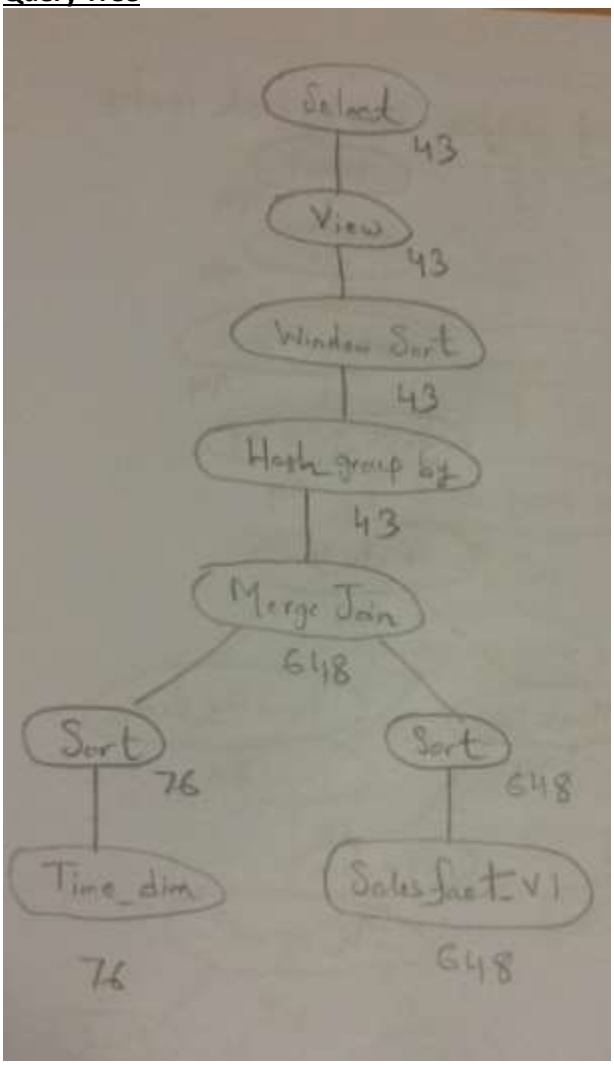
```
Plan hash value: 2428629332

-----------------------------------------------------------------------------------------------
| Id  | Operation              | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |               |    43 |  1419 |    11  (37)| 00:00:01 |
|*  1 |  VIEW                  |               |    43 |  1419 |    11  (37)| 00:00:01 |
|   2 |   WINDOW SORT          |               |    43 |  1333 |    11  (37)| 00:00:01 |
|   3 |    HASH GROUP BY       |               |    43 |  1333 |    11  (37)| 00:00:01 |
|   4 |     MERGE JOIN         |               |   648 | 20088 |     9  (23)| 00:00:01 |
|   5 |      SORT JOIN         |               |    76 |  1292 |     4  (25)| 00:00:01 |
|   6 |       TABLE ACCESS FULL| TIME_DIM      |    76 |  1292 |     3   (0)| 00:00:01 |
|*  7 |      SORT JOIN         |               |   648 |  9072 |     5  (20)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| SALES_FACT_V1 |   648 |  9072 |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result

	YEAR	MONTH	TOTAL_REVENUE	Percent Rank
1	2007	06	352186.8	0.91666666666666666666666666666666666666666667
2	2007	09	367945.9	0.95833333333333333333333333333333333333333333
3	2007	11	398487.1	1

Query Tree



COMPARISON: The first query uses hash join where else the second query uses sort merge which is less efficient than hash join.

REPORTS WITH PARTITIONS

--TOP 3 PRODUCTS BY EACH YEAR

FIRST QUERY (Q4)

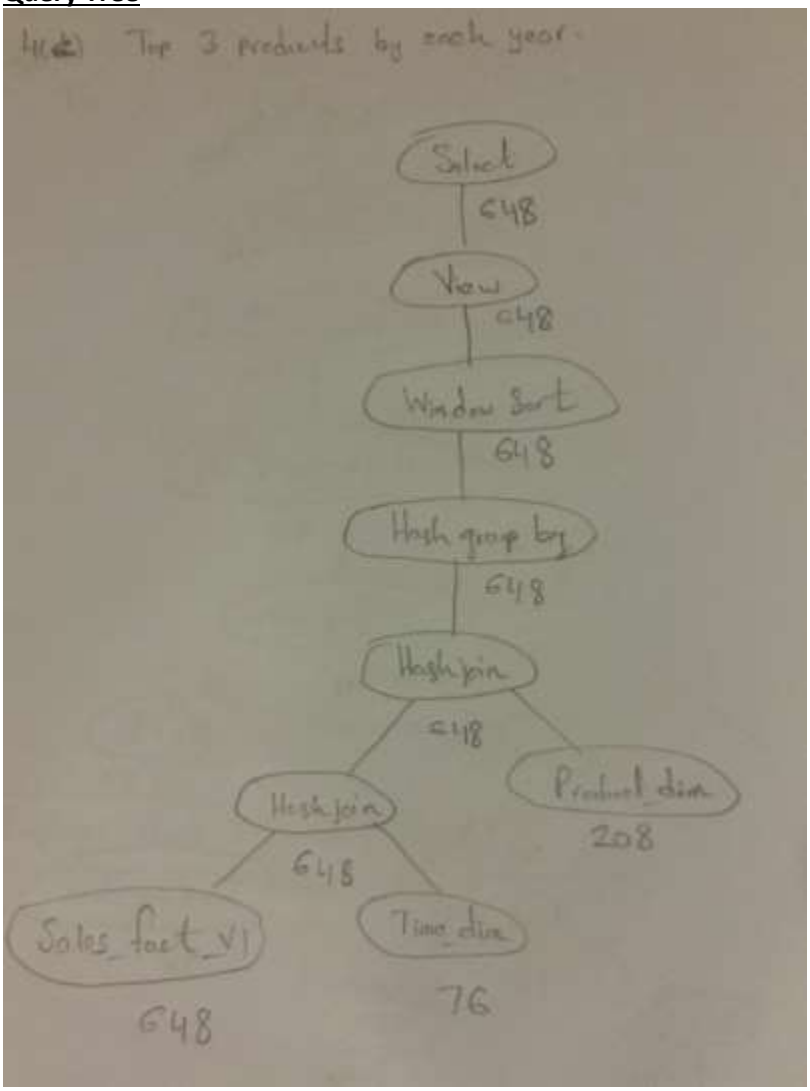
```
Plan hash value: 2542793835

-----------------------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |               |   648 | 36936 |    13  (16)| 00:00:01 |
|*  1 |  VIEW                    |               |   648 | 36936 |    13  (16)| 00:00:01 |
|*  2 |   WINDOW SORT PUSHED RANK|               |   648 | 33048 |    13  (16)| 00:00:01 |
|   3 |    HASH GROUP BY         |               |   648 | 33048 |    13  (16)| 00:00:01 |
|*  4 |     HASH JOIN            |               |   648 | 33048 |    11   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL   | PRODUCT_DIM   |   208 |  4368 |     4   (0)| 00:00:01 |
|*  6 |      HASH JOIN           |               |   648 | 19440 |     7   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL  | TIME_DIM      |    76 |  1064 |     3   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL  | SALES_FACT_V1 |   648 | 10368 |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
```

Query result

	YEAR	PRODUCT_NAME	TOTAL_ORDERS	ORDERS_RANK
58	2006	Project Management - S3.3	1	2
59	2006	Sound Card STD	1	2
60	2006	Business Cards - 1000/2L	1	2
61	2007	LaserPro 600/6/BW	12	1
62	2007	Screws <B.28.S>	11	2
63	2007	Mouse C/E	9	3
64	2008	Sound Card STD	6	1
65	2008	Screws <B.28.S>	5	2
66	2008	Mouse C/E	5	2
67	2008	LaserPro 600/6/BW	3	3
68	2008	Screws <S.16.S>	3	3
69	2008	Manual - Vision Tools2.0	3	3
70	2008	Manual - Vision OS/2.x	3	3

Query Tree


Top 3 products by each year
Select
648
View
648
Window Sort
648
Hash group by
648
Hash join
648
Product_dim
208
Hash join
648
Sales_fact_V1
648
Time_dim
76

SECOND QUERY (Q5)

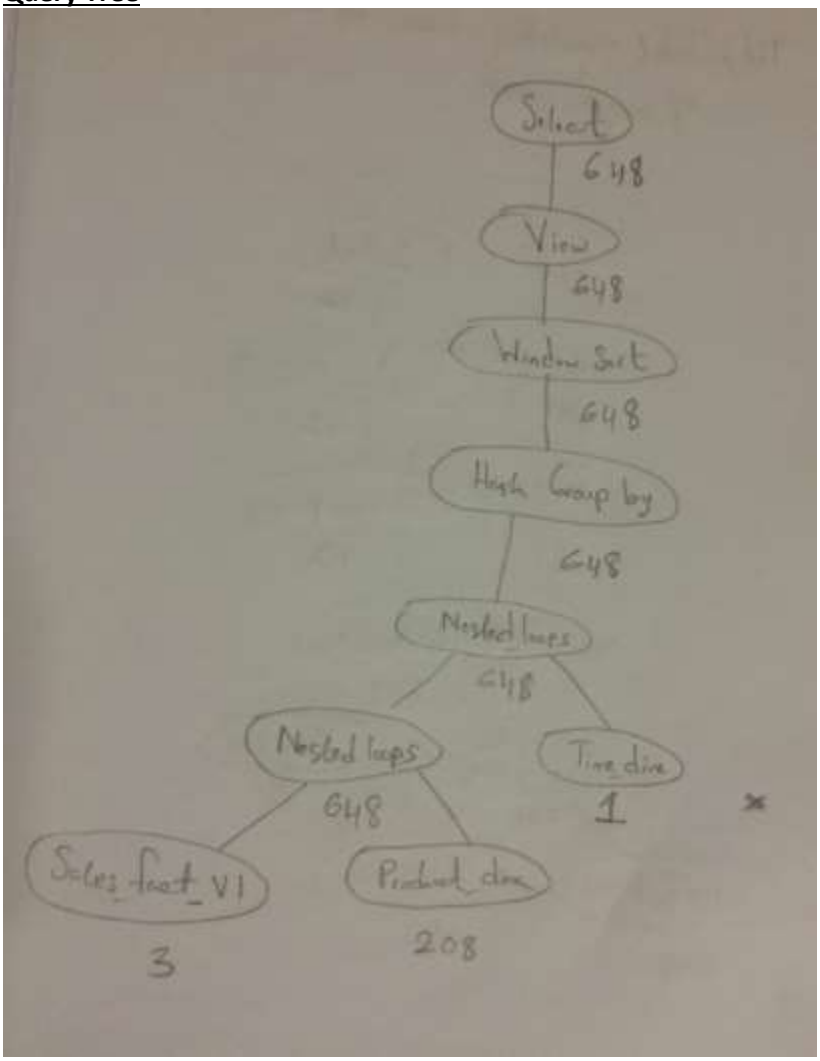
```
Plan hash value: 937073327

-------------------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |               |   648 | 36936 |  1107   (1)| 00:00:01 |
|*  1 |  VIEW                    |               |   648 | 36936 |  1107   (1)| 00:00:01 |
|*  2 |   WINDOW SORT PUSHED RANK|               |   648 | 33048 |  1107   (1)| 00:00:01 |
|   3 |    HASH GROUP BY         |               |   648 | 33048 |  1107   (1)| 00:00:01 |
|   4 |     NESTED LOOPS         |               |   648 | 33048 |  1105   (1)| 00:00:01 |
|   5 |      NESTED LOOPS        |               |   648 | 23976 |   402   (1)| 00:00:01 |
|   6 |       TABLE ACCESS FULL  | PRODUCT_DIM   |   208 |  4368 |     4   (0)| 00:00:01 |
|*  7 |       TABLE ACCESS FULL  | SALES_FACT_V1 |     3 |    48 |     2   (0)| 00:00:01 |
|*  8 |      TABLE ACCESS FULL   | TIME_DIM      |     1 |    14 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Query result

	YEAR	PRODUCT_NAME	TOTAL_ORDERS	ORDERS_RANK
58	2006	Project Management - S3.3	1	2
59	2006	Sound Card STD	1	2
60	2006	Business Cards - 1000/2L	1	2
61	2007	LaserPro 600/6/BW	12	1
62	2007	Screws <B.28.S>	11	2
63	2007	Mouse C/E	9	3
64	2008	Sound Card STD	6	1
65	2008	Screws <B.28.S>	5	2
66	2008	Mouse C/E	5	2
67	2008	LaserPro 600/6/BW	3	3
68	2008	Screws <S.16.S>	3	3
69	2008	Manual - Vision Tools2.0	3	3
70	2008	Manual - Vision OS/2.x	3	3

Query Tree



COMPARISON: The first query uses hash join where else the second query uses nested loops which is less efficient than hash join

------ Top 2 most employed jobs by each country

FIRST QUERY (Q4)

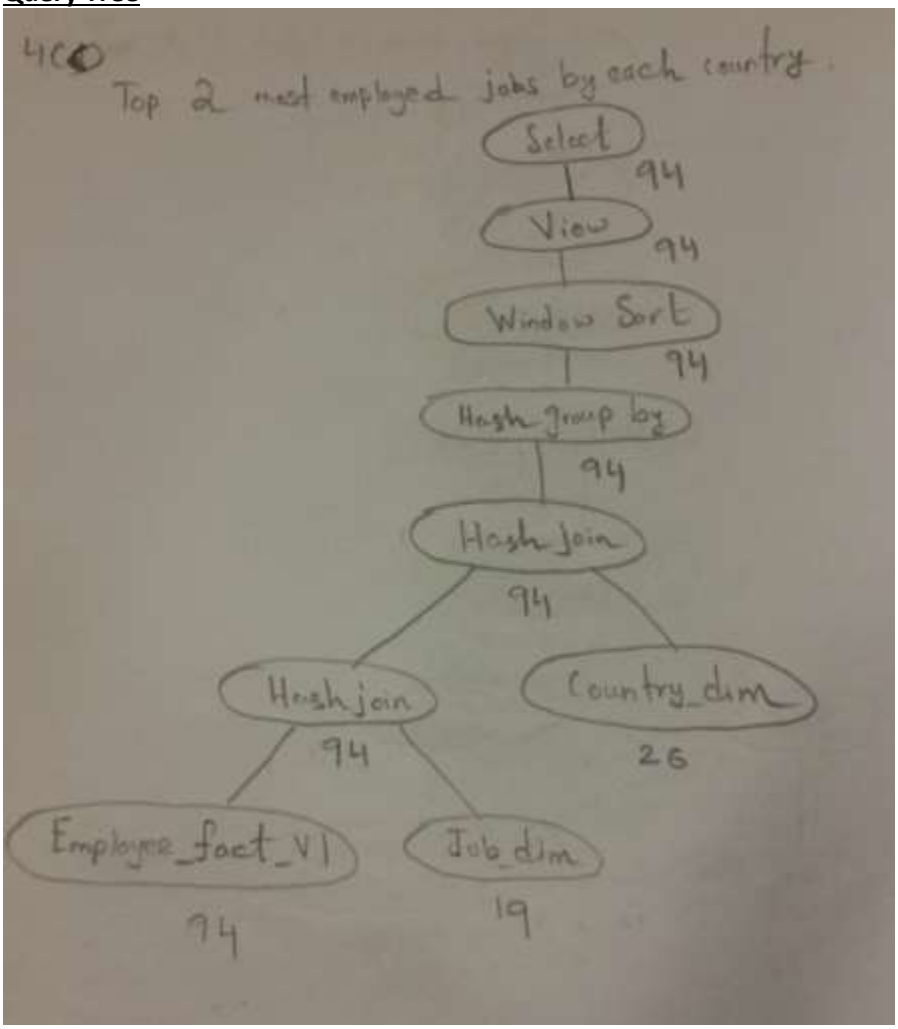
```
Plan hash value: 686207944

-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                  |    94 |  6298 |    11  (19)| 00:00:01 |
|*  1 |  VIEW                    |                  |    94 |  6298 |    11  (19)| 00:00:01 |
|*  2 |   WINDOW SORT PUSHED RANK|                  |    94 |  5076 |    11  (19)| 00:00:01 |
|   3 |    HASH GROUP BY         |                  |    94 |  5076 |    11  (19)| 00:00:01 |
|*  4 |     HASH JOIN            |                  |    94 |  5076 |     9   (0)| 00:00:01 |
|*  5 |      HASH JOIN           |                  |    94 |  3948 |     6   (0)| 00:00:01 |
|   6 |       TABLE ACCESS FULL  | JOB_DIM          |    19 |   513 |     3   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL  | EMPLOYEE_FACT_V1 |    94 |  1410 |     3   (0)| 00:00:01 |
|   8 |      TABLE ACCESS FULL   | COUNTRY_DIM      |    26 |   312 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result

	COUNTRY_NAME	JOB_TITLE	TOTAL_EMPLOYEES	RANK
1	Canada	Marketing Manager	1	1
2	Canada	Marketing Representative	1	1
3	Germany	Public Relations Representative	1	1
4	United Kingdom	Sales Representative	29	1
5	United Kingdom	Sales Manager	5	2
6	United States of America	Shipping Clerk	20	1
7	United States of America	Stock Clerk	20	1
8	United States of America	Accountant	5	2
9	United States of America	Purchasing Clerk	5	2
10	United States of America	Programmer	5	2
11	United States of America	Stock Manager	5	2

Query Tree


4C
Top 2 most employed jobs by each country.
Select
94
View
94
Window Sort
94
Hash group by
94
Hash join
94
Hash join
94
Country_dim
26
Employee_fact_V1
94
Job_dim
19

SECOND QUERY (Q5)

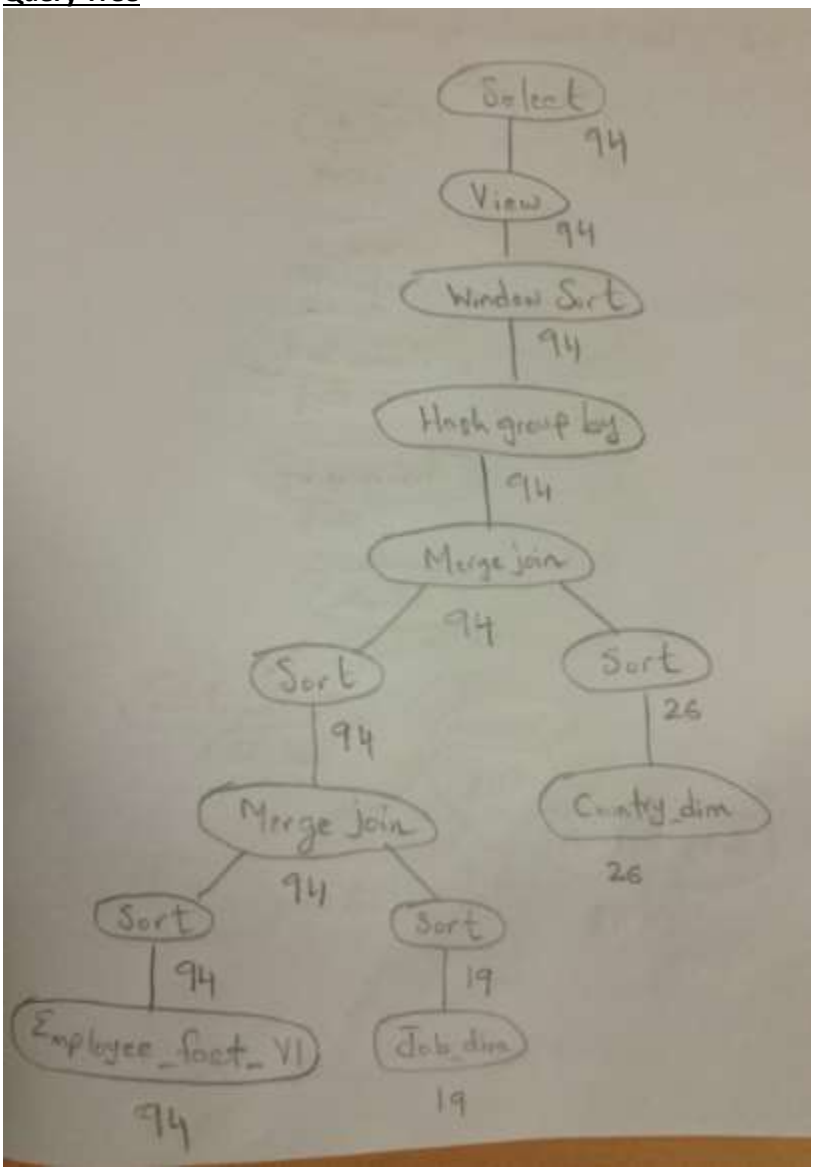
```
Plan hash value: 2092400436

-------------------------------------------------------------------------------------------------
| Id  | Operation                | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                  |    94 |  6298 |    15  (40)| 00:00:01 |
|*  1 |  VIEW                    |                  |    94 |  6298 |    15  (40)| 00:00:01 |
|*  2 |   WINDOW SORT PUSHED RANK|                  |    94 |  5076 |    15  (40)| 00:00:01 |
|   3 |    HASH GROUP BY         |                  |    94 |  5076 |    15  (40)| 00:00:01 |
|   4 |     MERGE JOIN           |                  |    94 |  5076 |    13  (31)| 00:00:01 |
|   5 |      SORT JOIN           |                  |    94 |  3948 |     9  (34)| 00:00:01 |
|   6 |       MERGE JOIN         |                  |    94 |  3948 |     8  (25)| 00:00:01 |
|   7 |        SORT JOIN         |                  |    19 |   513 |     4  (25)| 00:00:01 |
|   8 |         TABLE ACCESS FULL| JOB_DIM          |    19 |   513 |     3   (0)| 00:00:01 |
|*  9 |        SORT JOIN         |                  |    94 |  1410 |     4  (25)| 00:00:01 |
|  10 |         TABLE ACCESS FULL| EMPLOYEE_FACT_V1 |    94 |  1410 |     3   (0)| 00:00:01 |
|* 11 |      SORT JOIN           |                  |    26 |   312 |     4  (25)| 00:00:01 |
|  12 |       TABLE ACCESS FULL  | COUNTRY_DIM      |    26 |   312 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Query result

	COUNTRY_NAME	JOB_TITLE	TOTAL_EMPLOYEES	RANK
1	Canada	Marketing Manager	1	1
2	Canada	Marketing Representative	1	1
3	Germany	Public Relations Representative	1	1
4	United Kingdom	Sales Representative	29	1
5	United Kingdom	Sales Manager	5	2
6	United States of America	Shipping Clerk	20	1
7	United States of America	Stock Clerk	20	1
8	United States of America	Accountant	5	2
9	United States of America	Purchasing Clerk	5	2
10	United States of America	Programmer	5	2
11	United States of America	Stock Manager	5	2

Query Tree



COMPARISON: The first query uses hash join where else the second query uses sort merge which is less efficient than hash join

REPORTS WITH CUMULATIVE AND MOVING AGGREGATE

--total cumulative sales by each quarter of each year

FIRST QUERY (Q4)

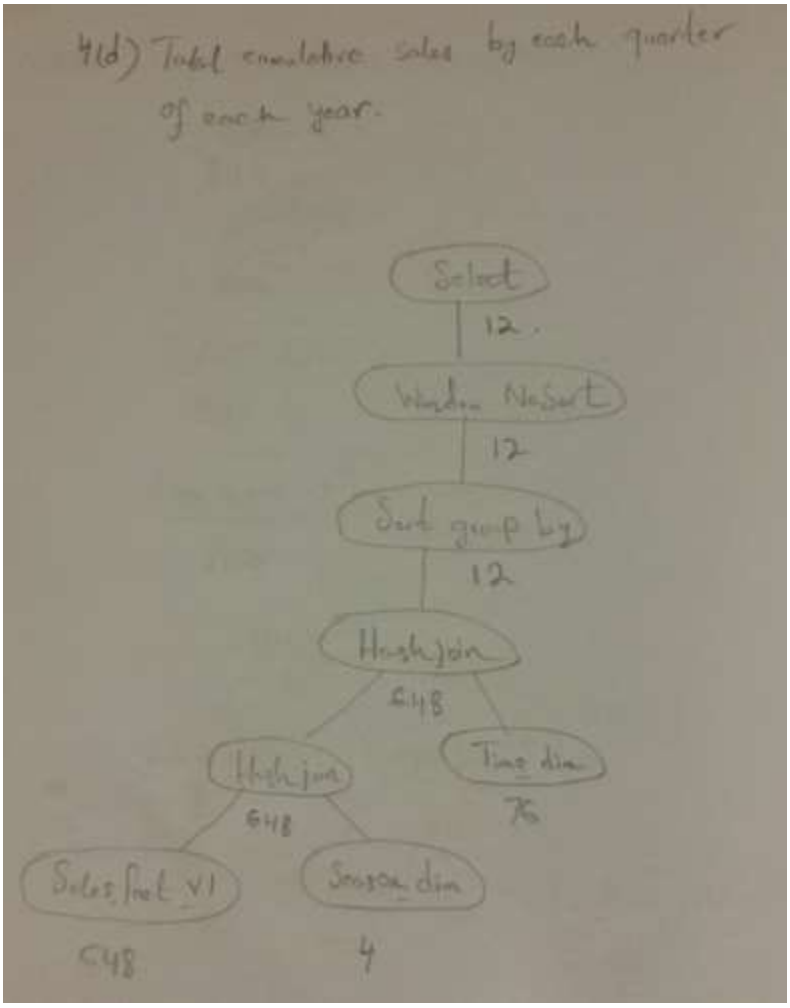
```
Plan hash value: 429249877

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               |    12 |   492 |    11  (10)| 00:00:01 |
|   1 |  WINDOW NOSORT        |               |    12 |   492 |    11  (10)| 00:00:01 |
|   2 |   SORT GROUP BY       |               |    12 |   492 |    11  (10)| 00:00:01 |
|*  3 |    HASH JOIN          |               |   648 | 26568 |    10   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL | TIME_DIM      |    76 |  1064 |     3   (0)| 00:00:01 |
|*  5 |     HASH JOIN         |               |   648 | 17496 |     7   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| SEASON_DIM    |     4 |    40 |     3   (0)| 00:00:01 |
|   7 |      TABLE ACCESS FULL| SALES_FACT_V1 |   648 | 11016 |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Query result

	SEASON_ID	SEASON_DESC	YEAR	TOTAL_SALES	CUM_SALES
1	1	Spring	2004	5,606	5,606
2	1	Spring	2006	385,702	391,308
3	2	Summer	2006	63,508	454,816
4	3	Autumn	2006	19,119	473,935
5	4	Winter	2006	12,543	486,478
6	1	Spring	2007	318,637	805,115
7	2	Summer	2007	801,119	1,606,234
8	3	Autumn	2007	709,316	2,315,549
9	4	Winter	2007	740,585	3,056,134
10	1	Spring	2008	141,048	3,197,183
11	2	Summer	2008	512,641	3,709,824
12	3	Autumn	2008	2,075	3,711,899

Query Tree


4(d) Total cumulative sales by each quarter of each year.
Select
12
Window NoSort
12
Sort group by
12
Hash join
648
Hash join
648
Time_dim
Sales_fact_V1
Season_dim
648
4

SECOND QUERY (Q5)

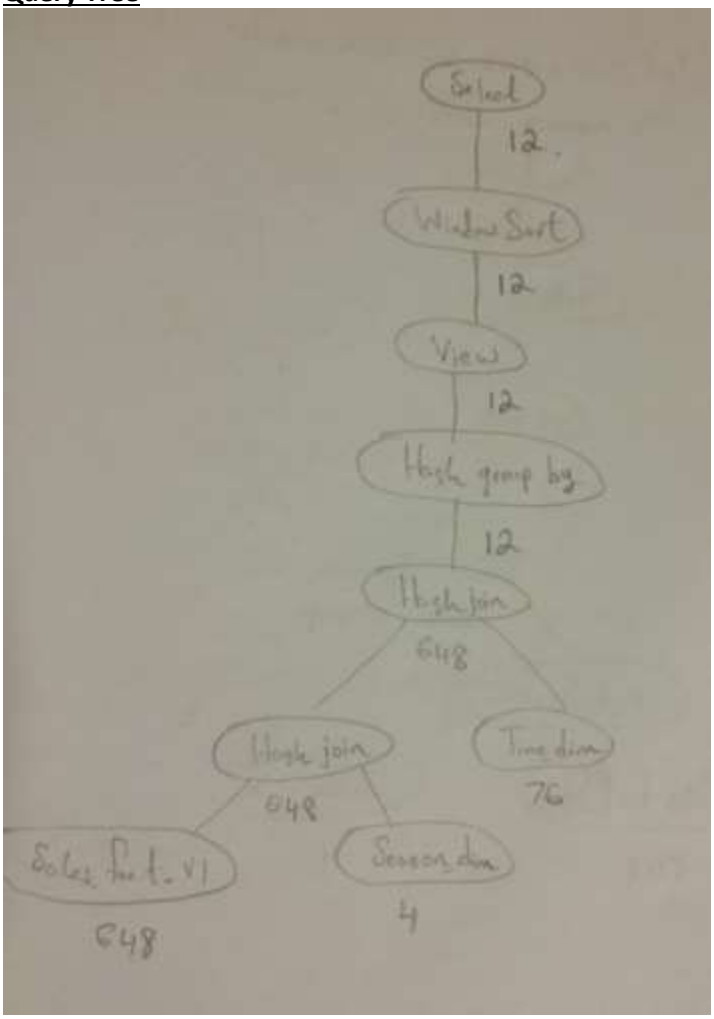
```
Plan hash value: 3969762636

-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |               |    12 |   444 |    12  (17)| 00:00:01 |
|   1 |  WINDOW SORT           |               |    12 |   444 |    12  (17)| 00:00:01 |
|   2 |   VIEW                 |               |    12 |   444 |    11  (10)| 00:00:01 |
|   3 |    HASH GROUP BY       |               |    12 |   492 |    11  (10)| 00:00:01 |
|*  4 |     HASH JOIN          |               |   648 | 26568 |    10   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL | TIME_DIM      |    76 |  1064 |     3   (0)| 00:00:01 |
|*  6 |      HASH JOIN         |               |   648 | 17496 |     7   (0)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| SEASON_DIM    |     4 |    40 |     3   (0)| 00:00:01 |
|   8 |       TABLE ACCESS FULL| SALES_FACT_V1 |   648 | 11016 |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Query result

	SEASON_ID	SEASON_DESC	YEAR	TOTAL_SALES	CUM_SALES
1	1	Spring	2004	5,606	5,606
2	1	Spring	2006	385,702	391,308
3	2	Summer	2006	63,508	454,816
4	3	Autumn	2006	19,119	473,935
5	4	Winter	2006	12,543	486,478
6	1	Spring	2007	318,637	805,115
7	2	Summer	2007	801,119	1,606,234
8	3	Autumn	2007	709,316	2,315,549
9	4	Winter	2007	740,585	3,056,134
10	1	Spring	2008	141,048	3,197,183
11	2	Summer	2008	512,641	3,709,824
12	3	Autumn	2008	2,075	3,711,899

Query Tree



COMPARISON: The second query first does the group by and then does the sorting but the first query does the opposite which is less efficient.

--total cumulative orders by each quarter of each year with moving window of 2 quarters

FIRST QUERY (Q4)

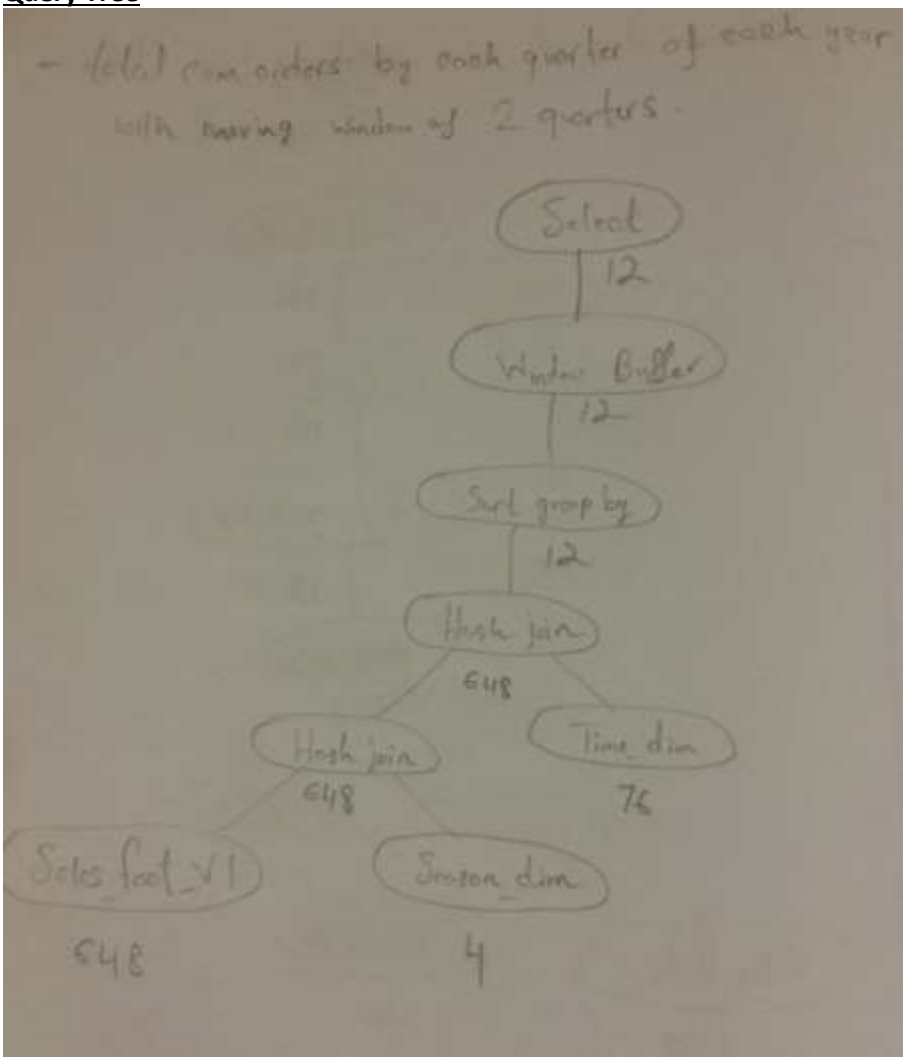
```
Plan hash value: 132471176

-----------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               |    12 |   468 |    11  (10)| 00:00:01 |
|   1 |  WINDOW BUFFER        |               |    12 |   468 |    11  (10)| 00:00:01 |
|   2 |   SORT GROUP BY       |               |    12 |   468 |    11  (10)| 00:00:01 |
|*  3 |    HASH JOIN          |               |   648 | 25272 |    10   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL | TIME_DIM      |    76 |  1064 |     3   (0)| 00:00:01 |
|*  5 |     HASH JOIN         |               |   648 | 16200 |     7   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| SEASON_DIM    |     4 |    40 |     3   (0)| 00:00:01 |
|   7 |      TABLE ACCESS FULL| SALES_FACT_V1 |   648 |  9720 |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Query result

	SEASON_ID	SEASON_DESC	YEAR	TOTAL_ORDERS	CUM_ORDERS
1	1	Spring	2004	6	6
2	1	Spring	2006	56	62
3	2	Summer	2006	23	85
4	3	Autumn	2006	12	91
5	4	Winter	2006	10	45
6	1	Spring	2007	68	90
7	2	Summer	2007	137	215
8	3	Autumn	2007	118	323
9	4	Winter	2007	103	358
10	1	Spring	2008	53	274
11	2	Summer	2008	85	241
12	3	Autumn	2008	2	140

Query Tree


- Select
with moving
Select
12
Window Buffer
12
Sort group by
12
Hash join
648
Hash join
648
Time_dim
76
Sales_fact_v1
648
Station_dim
4

SECOND QUERY (Q5)

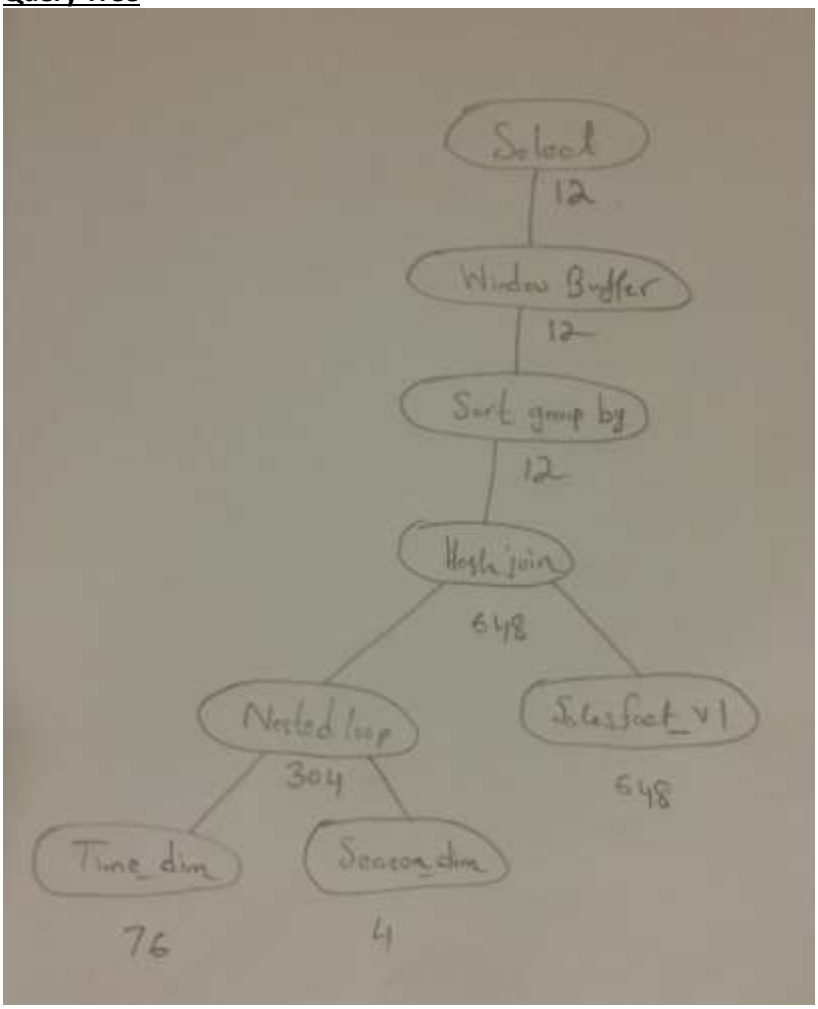
```
Plan hash value: 1706608709

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               |    12 |   468 |    14   (8)| 00:00:01 |
|   1 |  WINDOW BUFFER        |               |    12 |   468 |    14   (8)| 00:00:01 |
|   2 |   SORT GROUP BY       |               |    12 |   468 |    14   (8)| 00:00:01 |
|*  3 |    HASH JOIN          |               |   648 | 25272 |    13   (0)| 00:00:01 |
|   4 |     NESTED LOOPS      |               |   304 |  7296 |     9   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| SEASON_DIM    |     4 |    40 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| TIME_DIM      |    76 |  1064 |     2   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | SALES_FACT_V1 |   648 |  9720 |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Query result

	SEASON_ID	SEASON_DESC	YEAR	TOTAL_ORDERS	CUM_ORDERS
1	1	Spring	2004	6	6
2	1	Spring	2006	56	62
3	2	Summer	2006	23	85
4	3	Autumn	2006	12	91
5	4	Winter	2006	10	45
6	1	Spring	2007	68	90
7	2	Summer	2007	137	215
8	3	Autumn	2007	118	323
9	4	Winter	2007	103	358
10	1	Spring	2008	53	274
11	2	Summer	2008	85	241
12	3	Autumn	2008	2	140

Query Tree



COMPARISON: The second query uses nested loop which is less efficient compared to hash join which the first query uses.

ADDITIONAL CONTENT

After creating the sales fact table, I exported the data to Tableau (data visualization software).

Below is one of the visualizations that I created.