

Cover Sheet

FIT5195 S1 2021

FIT5195 Major Assignment

Tutorial Class ID, Day/Time:	5, Mon at 10:00	
Student ID	Surname	Other Name/s
31069282	Zemnukhov	Pavel
31187366	Jajoria	Prashant

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
31187366	Jajoria	Prashant
31069282	Zemnukhov	Pavel

* Please include the names of all other group members.

Unit name and code	FIT5195 Business intelligence and data warehousing - S1 2021	
Title of assignment	FIT5195 Major Assignment	
Lecturer/tutor	Assoc. Prof. David Taniar / Shuyi Sun, David Cheng Zarate, Joe Shao, Xiaojiao Du	
Tutorial day and time	05 Mon 10:00	Campus Clayton
Is this an authorised group assignment? <input checked="" type="checkbox"/> Yes <input type="checkbox"/> No		
Has any part of this assignment been previously submitted as part of another unit/course? <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No		
Due Date 5 May 2021, 11:55pm	Date submitted 5 May 2021	

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 The material used can be from any source (staff, students or the internet, published and unpublished works).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:
 - provide to another member of faculty and any external marker; and/or
 - submit it to a text matching software; and/or
 - 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 Date.....

* delete (iii) if not applicable

Signature Pavel Zemnukhov  Date: 05/05/2021

Signature Prashant Jajoria  Date: 05/05/2021

Signature _____ Date: _____ Signature _____ Date: _____

Signature _____ Date: _____ Signature _____ Date: _____

Privacy Statement

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

Contribution Declaration Form

(to be completed by all team members)

Please fill in the form with the contribution from each student towards the assignment.

1 NAME AND CONTRIBUTION DETAILS

Student ID	Student Name	Contribution Percentage
31187366	Prashant Jajoria	50%
31069282	Pavel Zemnukhov	50%

2 DECLARATION

We declare that:

- The information we have supplied in or with this form is complete and correct.
- We understand that the information we have provided in this form will be used for individual assessment of the assignment.

3 SIGNATURE

Signatures

Prashant Jajoria



Pavel Zemnukhov



Date

Day Month Year

05/05 / 2021

Details of your ORACLE accounts:

Username - S31069282

Password - Az123321

Username - S31187366

Password - Student

A contribution declaration form:

Percentage of contribution:

1. Name: Prashant Jajoria, ID: 31187366, Contribution: 50%

2. Name: Pavel Zemnukhov, ID: 31069282, Contribution: 50%

List of parts that each student did:

1. Prashant Jajoria:

Task C.1:

a- 50% of the ERD

b-errors 7-12

c- 50% of the design of Two versions of star/snowflake schema diagrams,

e-100%

Task C2:

a,c

Dimensions:

PROGRAM_DIM

PROGRAM_EVENT_HISTORY

PROGRAM_LENGTH_DIM

Facts:

ATTENDANCE_FACT_V1

b,c

Dimensions:

TOPIC_DIM

SUBSCRIPTION_DIM

MEDIA_DIM

REG_DIM

Facts:

ATTENDANCE_FACT_V1

INTEREST_FACT_V2

SUBSCRIPTION_FACT_V2

REGISTRATION_FACT_V2

2. Pavel Zemnukhov:

Task C.1:

Outputs:

a - 50% of the ERD

b - errors 1-6

c - 50% of the design of Two versions of star/snowflake schema diagrams,

d-100%

Task C.2:

Outputs:

a, c:

Dimensions:

TOPIC_DIM;

PROGRAM_DIM;

LOCATION_DIM;

MARITAL_STATUS_DIM;

OCCUPATION_DIM;

AGE_DIM;

PROGRAM_LENGTH_DIM;

TIME_DIM;

EVENT_SIZE_DIM;

TABLE MEDIA_DIM;

Facts:

SUBSCRIPTION_FACT_V1;

INTEREST_FACT_V1;

REGISTRATION_FACT_V1;

b,c:

Dimensions:

SUBSCRIPTION_DIM;

ADDRES_DIM;

EVENT_DIM;

ATTENDANCE_DIM;

Facts:

ATTENDANCE_FACT_V2;

C. Tasks

C.1

Design a data warehouse for the MonExplore database.

Preparation stage

a) The E/R diagram of the operational database:

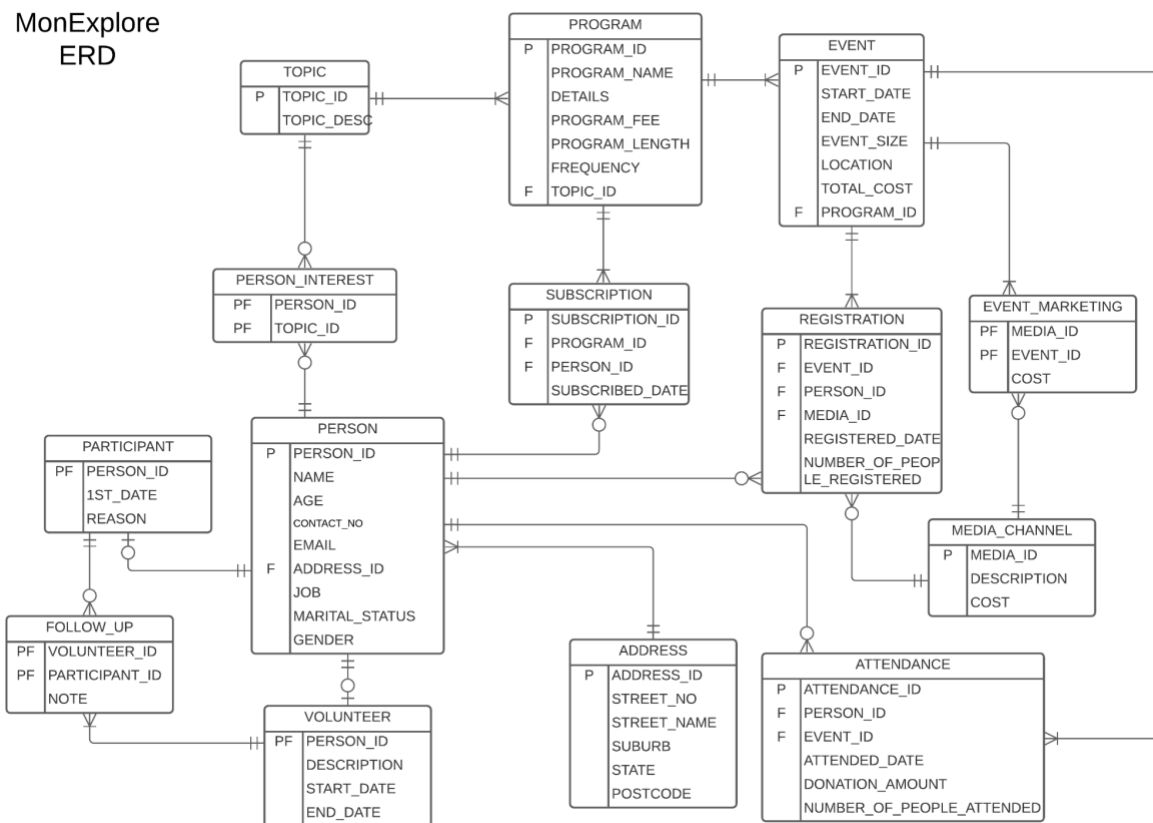


Figure 1 MonExplore ERD

Assumptions:

- More than 1 person can live at same address.
- There is at least 1 registration for an event.
- 3. Not every person is a participant.
- 4. Not every person is the volunteer
- 5. An event has at least 1 marketing channel.

b) Data cleaning:

Error 1:

EVENT Table

Checking event table for records, where the start date is greater than end date:

```
SELECT * FROM EVENT
```

```
WHERE TO_DATE(EVENT_START_DATE) > TO_DATE(EVENT_END_DATE);
```



SQL | All Rows Fetched: 2 in 11.757 seconds

	EVENT_ID	EVENT...	EVENT...	EVENT...	EVENT...	EVENT...	PROGR...
1	162	17.10.20	17.09.20	46	MonExplore	30	PR012
2	163	18.10.20	17.10.20	91	Online	40	PR007

Figure 2 EVENT table incorrect dates

Events start date greater than the vent end date, it is a mistake. We assume that it should be changed vice versa.

Update event 162:

```
UPDATE EVENT
```

```
SET EVENT_START_DATE = TO_DATE('17-SEP-2020', 'DD-MON-YYYY'),
```

```
EVENT_END_DATE = TO_DATE('17-OCT-2020', 'DD-MON-YYYY')
```

```
WHERE EVENT_ID = 162;
```


1 row updated.

Figure 3 Row update

Observe changes:

```
SELECT * FROM EVENT
```

```
WHERE EVENT_ID = 162;
```



SQL | All Rows Fetched: 1 in 0.017 seconds

	EVENT_ID	EVENT_STA...	EVENT_END...	EVENT_SIZE	EVENT_LOCATION	EVEN...	PROG...
1	162	17.09.20	17.10.20	46	MonExplore	30	PR012

Figure 4 Changed EVENT table

Update event 163:

```
UPDATE EVENT
```

```
SET EVENT_START_DATE = TO_DATE('17-OCT-2020', 'DD-MON-YYYY'),
```

```
EVENT_END_DATE = TO_DATE('18-OCT-2020', 'DD-MON-YYYY')
```

```
WHERE EVENT_ID = 163;
```

1 row updated.

Figure 5 Row update

Observe changes:

```
SELECT * FROM EVENT
```

```
WHERE EVENT_ID = 163;
```



SQL | All Rows Fetched: 1 in 0.013 seconds

	EVENT_ID	EVENT_START_DATE	EVENT_END_DATE	EVENT_SIZE	EVENT_LOCATION	EVENT_COST	PROGRAM_ID
1	163	17.10.20	18.10.20	91	Online	40	PR007

Figure 6 Changed EVENT table

Commit changes:
COMMIT;

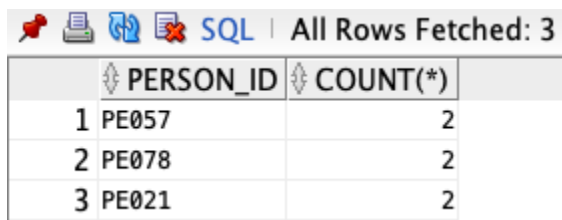
Commit complete.

Figure 7 Commit message**Error 2:**

PERSON table

Checking table for duplicate records:

```
SELECT PERSON_ID, COUNT(*)
FROM PERSON
GROUP BY PERSON_ID
HAVING COUNT(*) > 1;
```



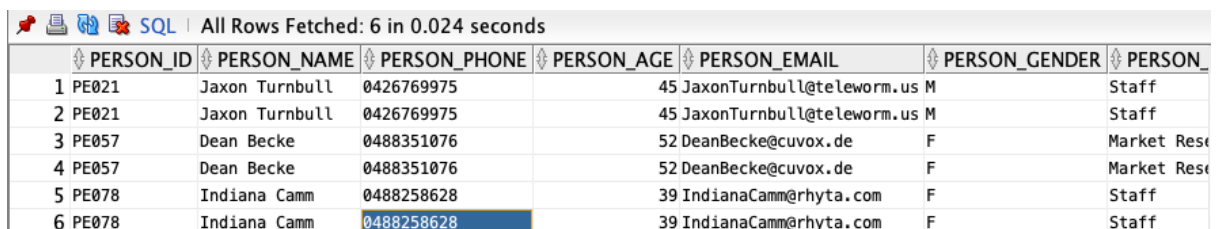
SQL | All Rows Fetched: 3

	PERSON_ID	COUNT(*)
1	PE057	2
2	PE078	2
3	PE021	2

Figure 8 PERSON table duplicate records

Looking for found duplicate records in table:

```
SELECT * FROM PERSON
WHERE PERSON_ID = 'PE057' OR PERSON_ID = 'PE078' OR PERSON_ID = 'PE021'
ORDER BY PERSON_ID;
```



SQL | All Rows Fetched: 6 in 0.024 seconds

	PERSON_ID	PERSON_NAME	PERSON_PHONE	PERSON_AGE	PERSON_EMAIL	PERSON_GENDER	PERSON_JOB
1	PE021	Jaxon Turnbull	0426769975	45	JaxonTurnbull@teleworm.us	M	Staff
2	PE021	Jaxon Turnbull	0426769975	45	JaxonTurnbull@teleworm.us	M	Staff
3	PE057	Dean Becke	0488351076	52	DeanBecke@cuvox.de	F	Market Res
4	PE057	Dean Becke	0488351076	52	DeanBecke@cuvox.de	F	Market Res
5	PE078	Indiana Camm	0488258628	39	IndianaCamm@rhyta.com	F	Staff
6	PE078	Indiana Camm	0488258628	39	IndianaCamm@rhyta.com	F	Staff

Figure 9 PERSON table duplicate records

Create a new table with distinct PK:

```
CREATE TABLE PERSON2 AS
SELECT DISTINCT *
FROM PERSON;
```

Table PERSON2 created.

Figure 10 Table creation message

Checking new table for duplicate records:


```
SELECT PERSON_ID, COUNT(*)
FROM PERSON2
GROUP BY PERSON_ID
HAVING COUNT(*) > 1;
```

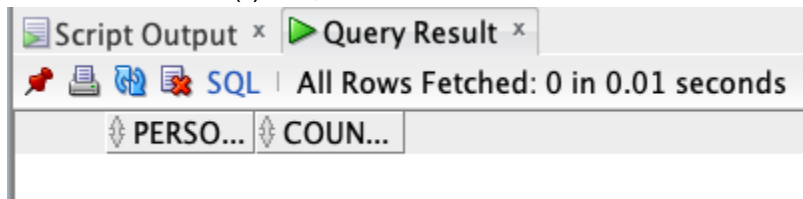


Figure 11 PERSON2 check for duplicate records

Dropping old table person:
DROP TABLE PERSON;

Table PERSON dropped.

Figure 12 Drop table message

Recreating table person with distinct PK:
CREATE TABLE PERSON AS
SELECT *
FROM PERSON2;
Table PERSON created.

Figure 13 Table creation message

Drop table person2:
DROP TABLE PERSON2;

Table PERSON2 dropped.

Figure 14 Drop table message

Table PERSON2 was copied back to PERSON to keep the original name of the table.

Error 3:
Table EVENT

Checking if event size contains out of range values:
SELECT * FROM EVENT
WHERE EVENT_SIZE < 0;

SQL All Rows Fetched: 2 in 0.015 seconds							
EVENT_ID	EVENT_START_DATE	EVENT_END_DATE	EVENT_SIZE	EVENT_LOCATION	EVENT_COST	PROGRAM_ID	
1	11 02.03.18	02.03.18	-10	MonExplore		0 PR007	
2	47 04.10.18	18.10.18	-75	Online		30 PR015	

Figure 15 EVENT table records with incorrect event_size

Comparing the values of event_size, we assume that it should only contain positive values and in a current case with negative values, it is a typo and requires changes to the positive.

Moreover, these events are involved in other tables, hence it is clear, that records required changes, not deletion.

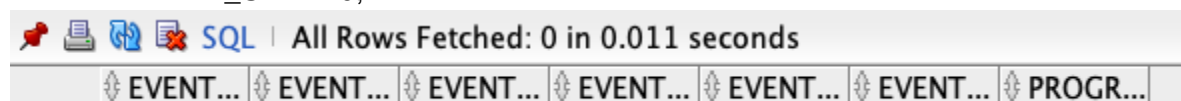
Updating event table:

```
UPDATE EVENT
SET EVENT_SIZE = EVENT_SIZE * (-1)
WHERE EVENT_SIZE < 0;
2 rows updated.
```

Figure 16 Update table message

Checking results:

```
SELECT * FROM EVENT
WHERE EVENT_SIZE < 0;
```



SQL | All Rows Fetched: 0 in 0.011 seconds

EVENT...	EVENT...	EVENT...	EVENT...	EVENT...	EVENT...	PROGR...
----------	----------	----------	----------	----------	----------	----------

Figure 17 EVENT table, checking results

Commit changes:

```
COMMIT;
Commit complete.
```

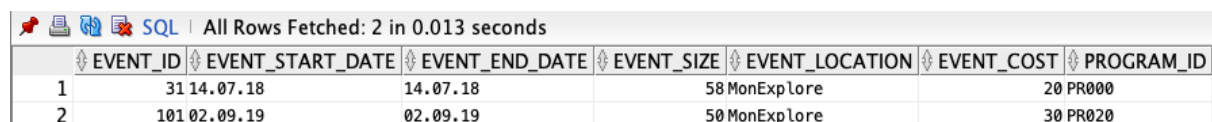
Figure 18 Commit message

Error 4:

Table EVENT

Checking invalid FK values (program_id):

```
SELECT * FROM EVENT
WHERE PROGRAM_ID NOT IN (
SELECT PROGRAM_ID FROM PROGRAM);
```



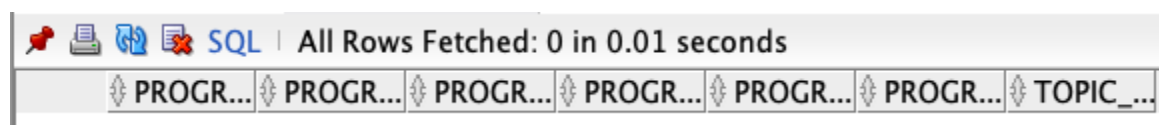
SQL | All Rows Fetched: 2 in 0.013 seconds

EVENT_ID	EVENT_START_DATE	EVENT_END_DATE	EVENT_SIZE	EVENT_LOCATION	EVENT_COST	PROGRAM_ID
1	31.07.18	14.07.18	58	MonExplore	20	PR000
2	101.09.19	02.09.19	50	MonExplore	30	PR020

Figure 19 EVENT table records with invalid FK values

Checking found invalid FK in PROGRAM table:

```
SELECT * FROM PROGRAM
WHERE PROGRAM_ID = 'PR000';
```



SQL | All Rows Fetched: 0 in 0.01 seconds

PROGR...	PROGR...	PROGR...	PROGR...	PROGR...	PROGR...	TOPIC_...
----------	----------	----------	----------	----------	----------	-----------

Figure 20 PROGRAM table, checking invalid FK

```
SELECT * FROM PROGRAM
```

WHERE PROGRAM_ID = 'PR020';

Script Output x Query Result x

SQL | All Rows Fetched: 0 in 0.01 seconds

PROGR...	PROGR...	PROGR...	PROGR...	PROGR...	PROGR...	TOPIC_...
----------	----------	----------	----------	----------	----------	-----------

Figure 21 PROGRAM table, checking invalid FK

Checking event_id with invalid FK in connected tables:

SELECT * FROM EVENT_MARKETING

WHERE EVENT_ID = 31 OR EVENT_ID = 101;

SQL | All Rows Fetched: 1 in 0.013 seconds

MEDIA_ID	EVENT_ID	EM_COST
1 MC004	101	100

Figure 22 EVENT_MARKETING table, checking events with incorrect FK

SELECT * FROM ATTENDANCE

WHERE EVENT_ID = 31 OR EVENT_ID = 101

ORDER BY EVENT_ID;

SQL | All Rows Fetched: 14 in 0.013 seconds

ATT_ID	ATT_DATE	ATT_DONATION_AMOUNT	ATT_NUM_OF_PEOPLE_ATTENDED	EVENT_ID	PERSON_ID
1	1360 14.07.18	20	3	31 PE096	
2	5136 14.07.18	65	10	31 PE092	
3	5627 14.07.18	20	10	31 PE038	
4	4832 14.07.18	15	3	31 PE061	
5	3325 14.07.18	55	3	31 PE041	
6	2425 14.07.18	50	2	31 PE089	
7	1827 14.07.18	20	8	31 PE041	
8	4178 02.09.19	75	10	101 PE026	
9	1263 02.09.19	15	6	101 PE080	
10	2577 02.09.19	10	6	101 PE094	
11	4766 02.09.19	25	7	101 PE061	
12	5410 02.09.19	5	4	101 PE052	
13	1791 02.09.19	20	4	101 PE061	
14	925 02.09.19	65	7	101 PE095	

Figure 23 ATTENDANCE table, checking events with incorrect FK

Due to the existing records in event_marketing and attendance tables for event_id = 31, event_id = 101, event records will not be deleted, but program_id will be set to null for the values, which are not presented in the program table.

Updating EVENT table:

UPDATE EVENT

SET PROGRAM_ID = NULL

WHERE PROGRAM_ID NOT IN (

SELECT PROGRAM_ID FROM PROGRAM);

2 rows updated.

Figure 24 Rows update message

Checking results:

```
SELECT * FROM EVENT
WHERE EVENT_ID = 31 OR EVENT_ID = 101;
```



EVENT_ID	EVENT_START_DATE	EVENT_END_DATE	EVENT_SIZE	EVENT_LOCATION	EVENT_COST	PROGRAM_ID
1	31 14.07.18	14.07.18	58	MonExplore	20	(null)
2	101 02.09.19	02.09.19	50	MonExplore	30	(null)

Figure 25 EVENT table, checking results after the update

```
SELECT * FROM EVENT
WHERE PROGRAM_ID NOT IN (
SELECT PROGRAM_ID FROM PROGRAM);
```



EVENT...	EVENT...	EVENT...	EVENT...	EVENT...	EVENT...	PROGR...
----------	----------	----------	----------	----------	----------	----------

Figure 26 EVENT table, checking FK

Commit changes:

```
COMMIT;
```

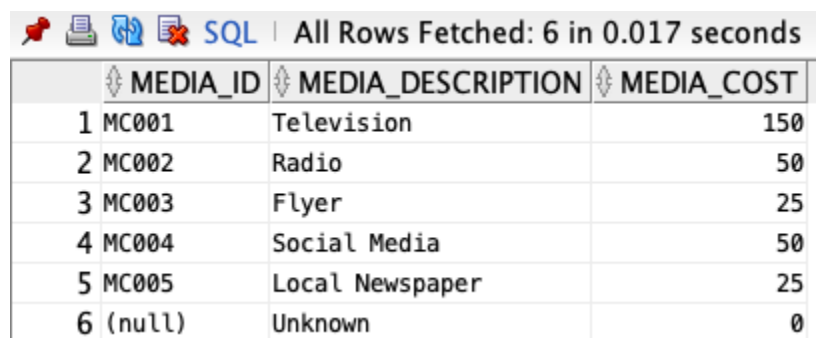
```
|Commit complete.
```

Figure 27 Commit message**Error 5:**

MEDIA_CHANNEL table

Observe table:

```
SELECT * FROM MEDIA_CHANNEL;
```



MEDIA_ID	MEDIA_DESCRIPTION	MEDIA_COST
1 MC001	Television	150
2 MC002	Radio	50
3 MC003	Flyer	25
4 MC004	Social Media	50
5 MC005	Local Newspaper	25
6 (null)	Unknown	0

Figure 28 MEDIA_CHANNEL table

Media_id is PK and cannot contain null values, because it is a unique identifier, therefore the record for the unknown media_description will be deleted.

Tables Registration, event marketing were checked, their records within media_id primary keys values of media_channel table, hence the record with null PK can be deleted.

Observe REGISTRATION table FK in MEDIA_CHANNEL:

```
SELECT * FROM REGISTRATION
WHERE MEDIA_ID NOT IN (SELECT MEDIA_ID FROM MEDIA_CHANNEL);
```



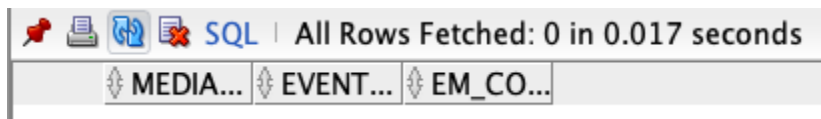
SQL | All Rows Fetched: 0 in 0.011 seconds

REG_ID	REG_N...	REG_D...	EVENT...	PERSO...	MEDIA...
--------	----------	----------	----------	----------	----------

Figure 29 REGISTRATION table, checking FK

Observe EVENT_MARKETING table FK in MEDIA_CHANNEL:

```
SELECT *
FROM EVENT_MARKETING
WHERE MEDIA_ID NOT IN
(SELECT MEDIA_ID FROM MEDIA_CHANNEL);
```



SQL | All Rows Fetched: 0 in 0.017 seconds

MEDIA...	EVENT...	EM_CO...
----------	----------	----------

Figure 30 EVENT_MARKETING table, checking FK

Delete incorrect PK:

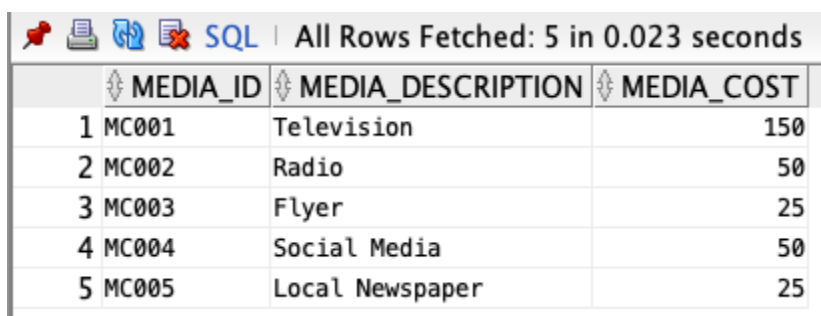
```
DELETE FROM MEDIA_CHANNEL
WHERE MEDIA_ID IS NULL;
```

1 row deleted.

Figure 31 Delete message

Observe changes:

```
SELECT * FROM MEDIA_CHANNEL;
```



SQL | All Rows Fetched: 5 in 0.023 seconds

MEDIA_ID	MEDIA_DESCRIPTION	MEDIA_COST
1 MC001	Television	150
2 MC002	Radio	50
3 MC003	Flyer	25
4 MC004	Social Media	50
5 MC005	Local Newspaper	25

Figure 32 MEDIA_CHANNEL table after changes

Error 6:
TOPIC table

Observe table:

```
SELECT * FROM TOPIC;
```

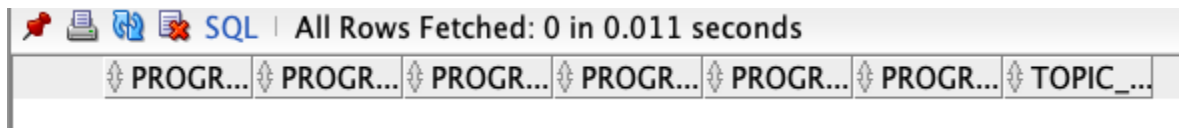
	TOPIC_ID	TOPIC_DESCRIPTION
1	T001	Networking
2	T002	Health and Lifestyle
3	T003	Spirituality
4	T004	Art and Culture
5	T005	Sport and Hobbies
6	T010	(null)

Figure 33 TOPIC table

Topic_id = T010 has a null value as a description, connected tables require checking.

Observe PROGRAM table:

```
SELECT * FROM PROGRAM
WHERE TOPIC_ID = 'T010';
```



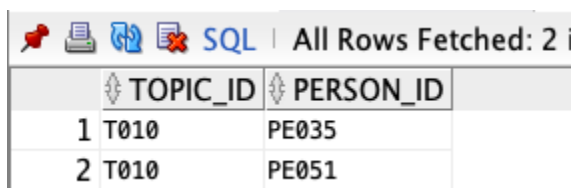
PROGR...	PROGR...	PROGR...	PROGR...	PROGR...	PROGR...	TOPIC_...
----------	----------	----------	----------	----------	----------	-----------

Figure 34 PROGRAM table, check for a topic with a null description

There are no programs for topic_id = 'T010', therefore this topic_id has no practical value in the database and can be deleted, but other connected tables need to be checked as well.

Observe PERSON_INTEREST table:

```
SELECT * FROM PERSON_INTEREST
WHERE TOPIC_ID = 'T010';
```



TOPIC_ID	PERSON_ID
1 T010	PE035
2 T010	PE051

Figure 35 PERSON_INTEREST table, check for a topic with a null description

Table person_interest has 2 values for topic_id = 'T010', but due to pointing to null topic_description and absence of programs for the topic, these 2 records can be deleted as well.

Delete null record from TOPIC:

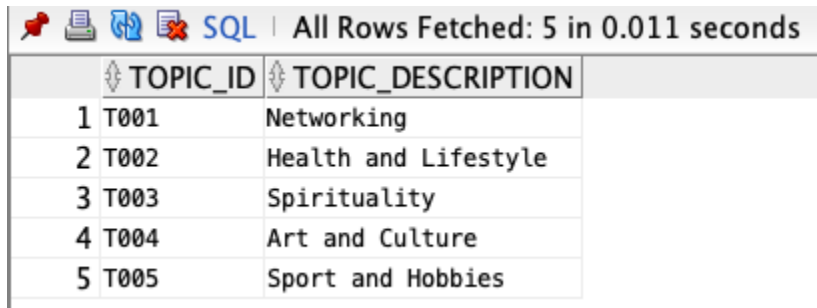
```
DELETE FROM TOPIC WHERE TOPIC_ID = 'T010';
```

1 row deleted.

Figure 36 Delete message

Observe TOPIC:

```
SELECT * FROM TOPIC;
```



SQL | All Rows Fetched: 5 in 0.011 seconds

	TOPIC_ID	TOPIC_DESCRIPTION
1	T001	Networking
2	T002	Health and Lifestyle
3	T003	Spirituality
4	T004	Art and Culture
5	T005	Sport and Hobbies

Figure 37 TOPIC table after changes

Delete records from PERSON_INTEREST:
 DELETE FROM PERSON_INTEREST
 WHERE TOPIC_ID = 'T010';

2 rows deleted.

Figure 38 Delete message

Observe PERSON_INTEREST:
 SELECT * FROM PERSON_INTEREST
 WHERE TOPIC_ID = 'T010';



SQL | All Rows Fetched: 0

	TOPIC_...	PERSO...
--	-----------	----------

Figure 39 PERSON_INTEREST table after changes

Commit changes:
 COMMIT;

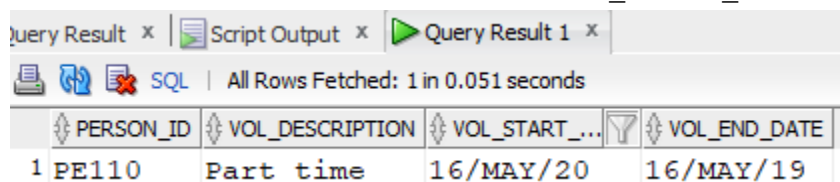
Commit complete.

Figure 40 Commit message

Error 7:

Check the Volunteer table to see if the start end for a volunteer is before the start date of volunteering.

SELECT * FROM VOLUNTEER WHERE VOL_START_DATE > VOL_END_DATE ;



Query Result x | Script Output x | Query Result 1 x

SQL | All Rows Fetched: 1 in 0.051 seconds

	PERSON_ID	VOL_DESCRIPTION	VOL_START_...	VOL_END_DATE
1	PE110	Part time	16/MAY/20	16/MAY/19

Figure 41 Select query results from Volunteer table

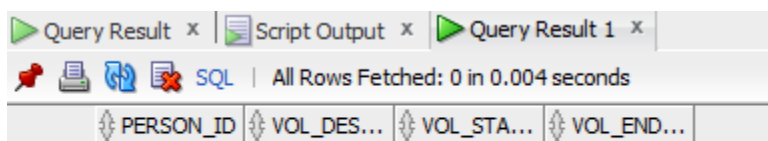
We find 1 volunteer data that has an incorrect start date and end date. Looking at the results we can see that the start and end date are misplaced and thus we swap the dates.

Update the volunteer start date and end date for a volunteer with person id = PE110.

```
UPDATE VOLUNTEER
SET VOL_START_DATE = VOL_END_DATE,
VOL_END_DATE = VOL_START_DATE
WHERE VOL_START_DATE > VOL_END_DATE ;
```

Checking the table for the update.

```
SELECT * FROM VOLUNTEER WHERE VOL_START_DATE > VOL_END_DATE ;
```



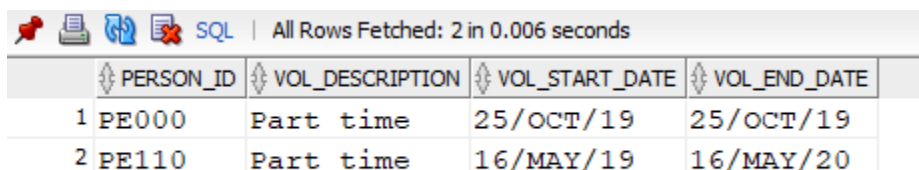
	PERSON_ID	VOL_DES...	VOL_STA...	VOL_END...
--	-----------	------------	------------	------------

Figure 42 Select query results from Volunteer table

Error 8:

As we store the Volunteer details also in the Person table, there should be a person_id for each of the Volunteer. Checking the Volunteer table to see if this condition is met.

```
SELECT * FROM VOLUNTEER
WHERE PERSON_ID NOT IN
(SELECT DISTINCT PERSON_ID FROM PERSON);
```



	PERSON_ID	VOL_DESCRIPTION	VOL_START_DATE	VOL_END_DATE
1	PE000	Part time	25/OCT/19	25/OCT/19
2	PE110	Part time	16/MAY/19	16/MAY/20

Figure 43 Select query results from Volunteer table

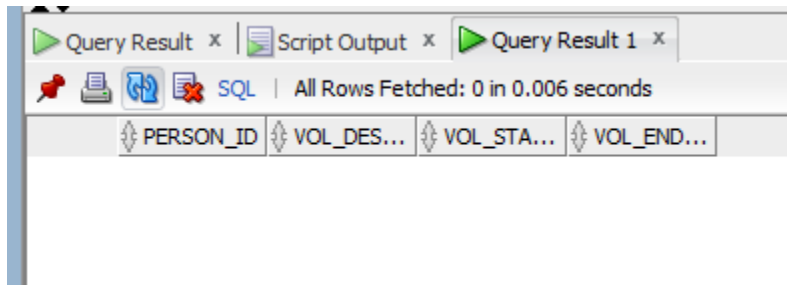
We find 2 Volunteer records who do not have a record in the Person table.

As all the Volunteer must have records in the Person table, we discard the Volunteer details of the volunteer with Person_id = PE000 and PE110.

```
DELETE FROM VOLUNTEER
WHERE PERSON_ID NOT IN
(SELECT DISTINCT PERSON_ID FROM PERSON);
```


Checking the table after deleting the 2 records from the volunteer table.

```
SELECT * FROM VOLUNTEER
WHERE PERSON_ID NOT IN
(SELECT DISTINCT PERSON_ID FROM PERSON);
```



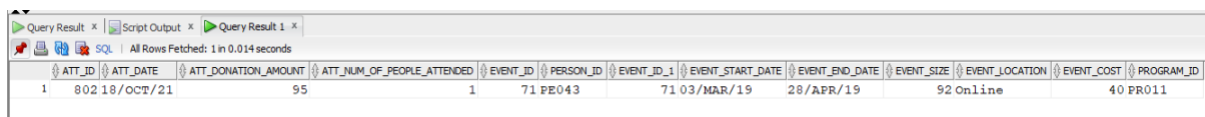
PERSON_ID	VOL_DES...	VOL_STA...	VOL_END...
-----------	------------	------------	------------

Figure 44 Select query results from Volunteer table

Error 9

Checking the Attendance table to see if there are any invalid records for attendance. We check to see if any records have an attendance date after the event has ended.

```
SELECT * FROM ATTENDANCE AE
JOIN EVENT EV
ON EV.EVENT_ID = AE.EVENT_ID
WHERE EV.EVENT_END_DATE < AE.ATT_DATE;
```



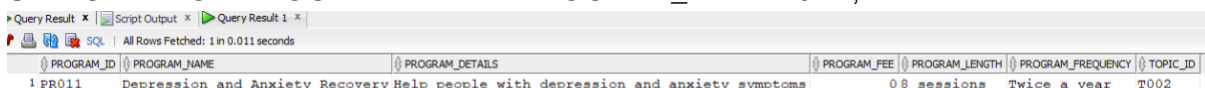
ATT_ID	ATT_DATE	ATT_DONATION_AMOUNT	ATT_NUM_OF_PEOPLE_ATTENDED	EVENT_ID	PERSON_ID	EVENT_ID_1	EVENT_START_DATE	EVENT_END_DATE	EVENT_SIZE	EVENT_LOCATION	EVENT_COST	PROGRAM_ID
1	802 18/OCT/21		95	1	71 PE043	71	03/MAR/19	28/APR/19	92	Online	40	PR011

Figure 45 Select query results from the Attendance table

We can see that the attendance record with id = 802 has an attendance date 18/OCT/21 for program id = PR011. The join between the Event and attendance table reveals that the Event has ended and the attendance is invalid.

We further check the Program table to see the number of sessions of Program PR011.

```
SELECT * FROM PROGRAM WHERE PROGRAM_ID = 'PR011';
```



PROGRAM_ID	PROGRAM_NAME	PROGRAM_DETAILS	PROGRAM_FEE	PROGRAM_LENGTH	PROGRAM_FREQUENCY	TOPIC_ID
1 PR011	Depression and Anxiety Recovery	Help people with depression and anxiety symptoms		08 sessions	Twice a year	T002

Figure 46 Select query results from Program table

As can be seen from the query result we can see that program PR011 has 8 sessions.

We further investigate to see the attendance of person PE043 for event 71.

```
SELECT * FROM ATTENDANCE WHERE EVENT_ID = '71' AND PERSON_ID = 'PE043';
```

	ATT_ID	ATT_DATE	ATT_DONATION_AMOUNT	ATT_NUM_OF_PEOPLE_ATTENDED	EVENT_ID	PERSON_ID
1	797	10/MAR/19	10	6	71	PE043
2	798	17/MAR/19	30	2	71	PE043
3	799	24/MAR/19	45	4	71	PE043
4	800	31/MAR/19	5	2	71	PE043
5	801	07/APR/19	20	6	71	PE043
6	802	18/OCT/21	95	1	71	PE043
7	803	21/APR/19	70	3	71	PE043
8	804	28/APR/19	5	8	71	PE043

Figure 47 Select query results from the Attendance table

We can see from the query result that the sessions are run every 7 days and thus the attendance date was incorrectly recorded for attendance id 802.

We change the attendance date to 14/APR/19 for attendance record no 802.

```
UPDATE ATTENDANCE
SET ATT_DATE = TO_DATE('14-APR-2019', 'DD-MON-YYYY')
WHERE ATT_ID = 802;
```

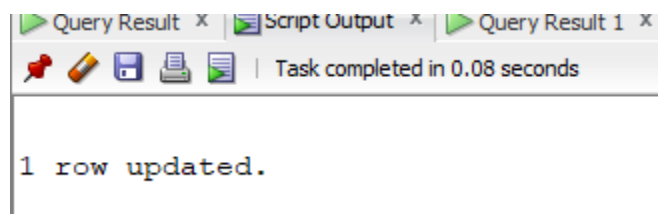


Figure 48 Update table message

Checking the attendance table after the update.

```
SELECT * FROM ATTENDANCE AE
JOIN EVENT EV
ON EV.EVENT_ID = AE.EVENT_ID
WHERE EV.EVENT_END_DATE < AE.ATT_DATE;
```

The screenshot shows a database interface with a 'Query Result' tab. Below the tab, there are icons for a pin, a pencil, a save icon, a printer, and a document. A status bar indicates 'All Rows Fetched: 0 in 0.018 seconds'. The main area displays a table with the following columns: ATT_ID, ATT_DATE, ATT_DONATION_AMOUNT, ATT_NUM_OF_PEOPLE_ATTENDED, EVENT_ID, PERSON_ID, EVENT_ID_1, EVENT_START_DATE, EVENT_END_DATE, EVENT_SIZE, EVENT_LOCATION, EVENT_CATEGORY, and PROGRAM_ID.

Figure 49 Select query results from the Attendance table

Error 10

We assume that there can be no registration for an event after it has commenced. We invalidate the corresponding attendance for such registrations.

BEFORE

These are the registrations for the events that have already started.

```

SELECT * FROM REGISTRATION RE
JOIN EVENT EV
ON RE.EVENT_ID = EV.EVENT_ID
WHERE RE.REG_DATE > EV.EVENT_START_DATE;

```

	REG_ID	REG_NUM_OF_PEOPLE_REGISTERED	REG_DATE	EVENT_ID	PERSON_ID	MEDIA_ID	EVENT_ID_1	EVENT_START_DATE	EVENT_END_DATE	EVENT_SIZE	EVENT_LOCATION	EVENT_COST	PROGRAM_ID
1	1023		2 10/OCT/20	162 PE098	MC004		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	
2	12		2 10/OCT/20	162 PE009	MC002		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	
3	296		1 10/OCT/20	162 PE013	MC004		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	
4	348		1 10/OCT/20	162 PE060	MC002		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	
5	1077		3 10/OCT/20	162 PE086	MC001		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	
6	1386		3 10/OCT/20	162 PE006	MC005		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	
7	1453		2 10/OCT/20	162 PE089	MC005		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	
8	1457		3 10/OCT/20	162 PE032	MC004		162 17/SEP/20	17/OCT/20		46 MonExplore		30 PR012	

Figure 50 Select query results from Registration and Event table joined

Corresponding attendance for the event registration.

```

SELECT * FROM ATTENDANCE
WHERE (PERSON_ID, EVENT_ID) IN (
    SELECT RE.PERSON_ID, RE.EVENT_ID FROM REGISTRATION RE
    JOIN EVENT EV
    ON RE.EVENT_ID = EV.EVENT_ID
    WHERE RE.REG_DATE > EV.EVENT_START_DATE
);

```

	ATT_ID	ATT_DATE	ATT_DONATION_AMOUNT	ATT_NUM_OF_PEOPLE_ATTENDED	EVENT_ID	PERSON_ID
1	57	17/OCT/20	15	3	162 PE009	
2	1195	17/OCT/20	15	10	162 PE013	
3	1399	17/OCT/20	20	8	162 PE060	
4	3925	17/OCT/20	60	5	162 PE098	
5	5476	17/OCT/20	95	1	162 PE089	
6	5481	17/OCT/20	15	4	162 PE032	
7	5212	17/OCT/20	65	4	162 PE006	
8	4135	17/OCT/20	55	2	162 PE086	

Figure 51 Select query results from the Attendance table

AFTER:

Delete the attendance as the registrations are invalid for the program.

```

DELETE FROM ATTENDANCE ATT
WHERE (ATT.PERSON_ID, ATT.EVENT_ID) IN (
    SELECT RE.PERSON_ID, RE.EVENT_ID FROM REGISTRATION RE
    JOIN EVENT EV
    ON RE.EVENT_ID = EV.EVENT_ID
    WHERE RE.REG_DATE > EV.EVENT_START_DATE
);

```

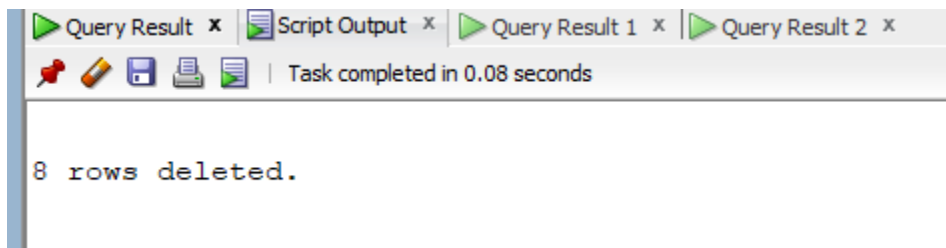


Figure 52 Delete rows message

Delete the registrations.

```
DELETE FROM REGISTRATION WHERE REG_ID IN (
SELECT REG_ID FROM REGISTRATION RE
JOIN EVENT EV
ON RE.EVENT_ID = EV.EVENT_ID
WHERE RE.REG_DATE > EV.EVENT_START_DATE);
```

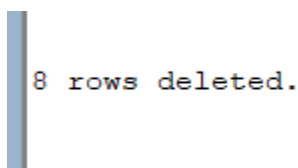


Figure 53 Delete rows message

Error 11

We check the donation amount where it is valid or not.

```
SELECT * FROM ATTENDANCE
WHERE ATT_DONATION_AMOUNT < 0;
```

 A screenshot of a SQL IDE interface. The top bar shows tabs for 'Script Output' and 'Query Result'. Below the tabs, a status bar indicates 'All Rows Fetched: 2 in 0.031 seconds'. The main area displays a table with two rows of data.

	ATT_ID	ATT_DATE	ATT_DONATION_AMOUNT	ATT_NUM_OF_PEOPLE_ATTENDED	EVENT_ID	PERSON_ID
1	639	12/NOV/20	-25	4	159	PE006
2	1001	28/MAY/19	-5	9	72	PE031

Figure 54 Select query results from the Attendance table

Updating the donation amount to keep it positive assuming it to be a data entry error.

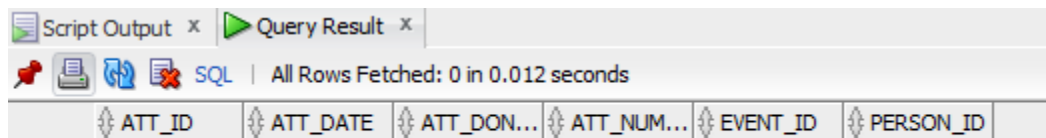
```
UPDATE ATTENDANCE ATT
SET ATT.ATT_DONATION_AMOUNT = ATT.ATT_DONATION_AMOUNT*(-1)
WHERE ATT_ID IN (SELECT ATT_ID FROM ATTENDANCE
WHERE ATT_DONATION_AMOUNT < 0);
```

```
2 rows updated.
```

Figure 55 Update table message

Checking the donation amount after the update.

```
SELECT * FROM ATTENDANCE
WHERE ATT_DONATION_AMOUNT < 0;
```



The screenshot shows a SQL query result window with the title 'Query Result'. The status bar indicates 'All Rows Fetched: 0 in 0.012 seconds'. The table has columns: ATT_ID, ATT_DATE, ATT_DONATION_AMOUNT, ATT_NUM_ROWS, EVENT_ID, and PERSON_ID. No rows are displayed.

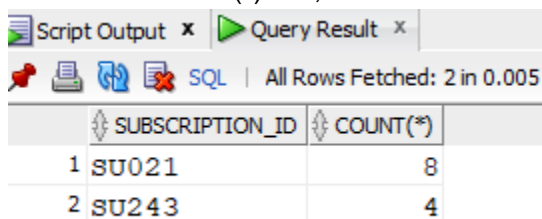
ATT_ID	ATT_DATE	ATT_DONATION_AMOUNT	ATT_NUM_ROWS	EVENT_ID	PERSON_ID
--------	----------	---------------------	--------------	----------	-----------

Figure 56 Select query results from the Attendance table

ERROR 12

We have duplicate rows with duplicate subscription_id in the subscription table. This is a violation of the Primary key constraint.

```
SELECT SUBSCRIPTION_ID, COUNT(*)
FROM SUBSCRIPTION
GROUP BY SUBSCRIPTION_ID
HAVING COUNT(*) > 1;
```



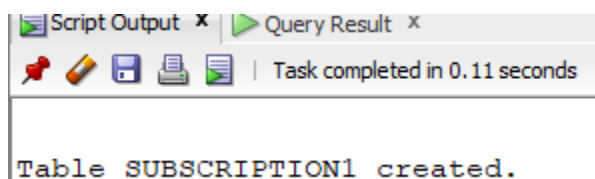
The screenshot shows a SQL query result window with the title 'Query Result'. The status bar indicates 'All Rows Fetched: 2 in 0.005'. The table has columns: SUBSCRIPTION_ID and COUNT(*). Two rows are displayed: 1 SU021 with a count of 8, and 2 SU243 with a count of 4.

SUBSCRIPTION_ID	COUNT(*)
1 SU021	8
2 SU243	4

Figure 57 Select query results from the Subscription table

Create a new Subscription table without the duplicated rows.

```
CREATE TABLE SUBSCRIPTION1 AS
SELECT DISTINCT * FROM SUBSCRIPTION;
```



The screenshot shows a SQL query result window with the title 'Query Result'. The status bar indicates 'Task completed in 0.11 seconds'. The message 'Table SUBSCRIPTION1 created.' is displayed.

Message
Table SUBSCRIPTION1 created.

Figure 58 Table creation message

We delete the old Subscription table, to have the new table take the same name as the old table.

```
DROP TABLE SUBSCRIPTION;
```

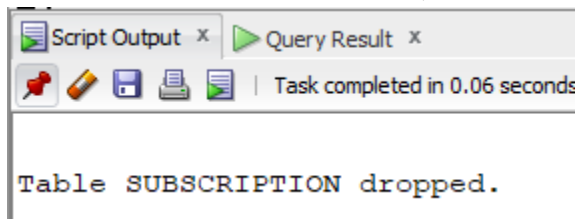


Figure 59 Table drop message

Create a new table that has the name Subscription from the updated table.

```
CREATE TABLE SUBSCRIPTION AS  
SELECT DISTINCT * FROM SUBSCRIPTION1;
```

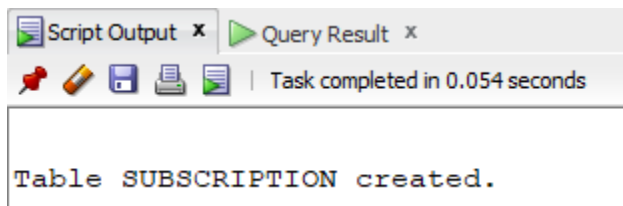


Figure 60 Table creation message

We check the Subscription table after creating a new table from the updated Subscription1 table.

```
SELECT SUBSCRIPTION_ID, COUNT(*)  
FROM SUBSCRIPTION  
GROUP BY SUBSCRIPTION_ID  
HAVING COUNT(*) > 1;
```

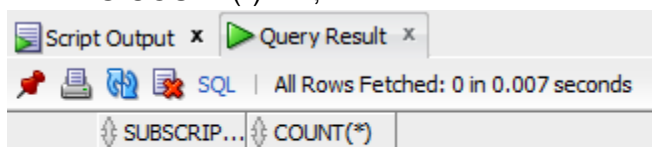


Figure 61 Select query results from Subscription table

Designing the data warehouse by drawing star/snowflake schema.

c) Two versions of star/snowflake schema diagrams

LEVEL 2

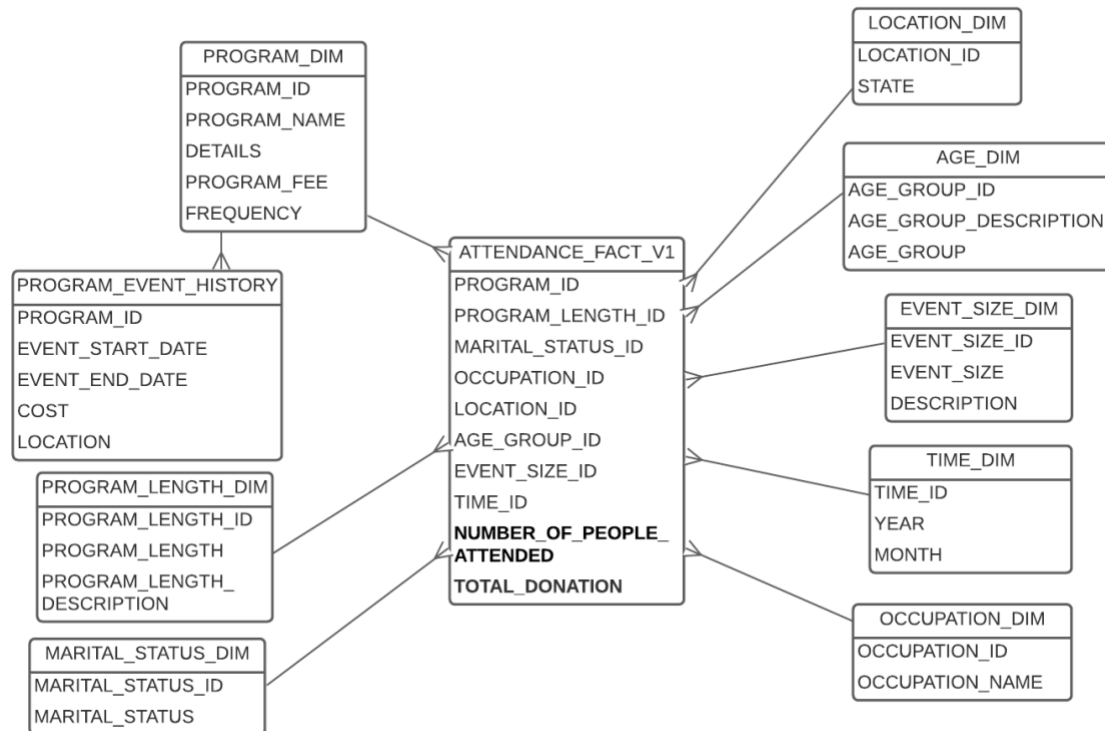


Figure 62: ATTENDANCE_FACT_V1

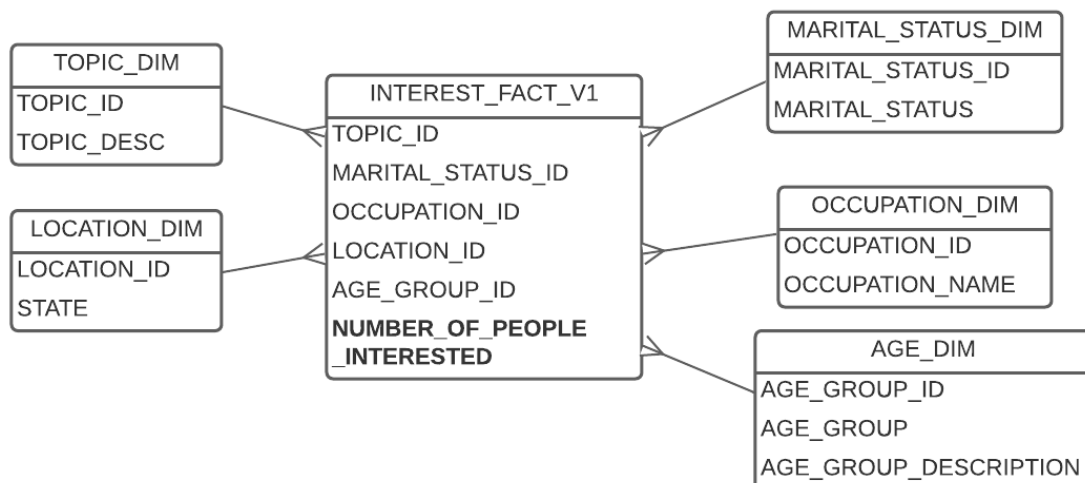


Figure 63 : INTEREST_FACT_V1

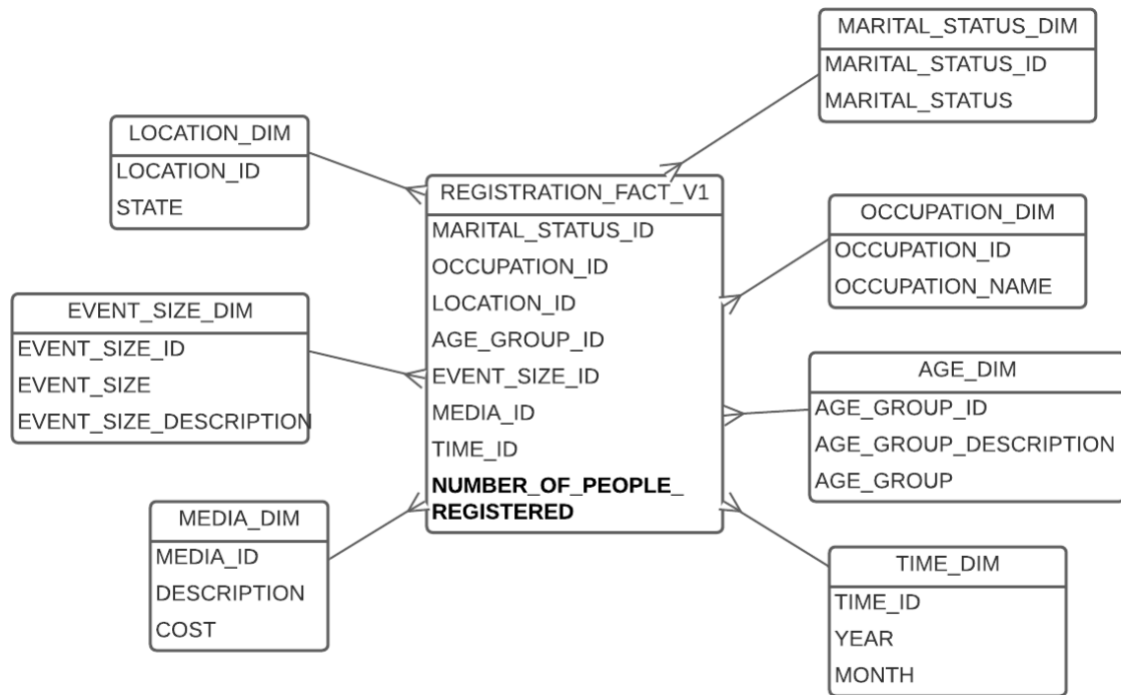


Figure 64: REGISTRATION_FACT_V1

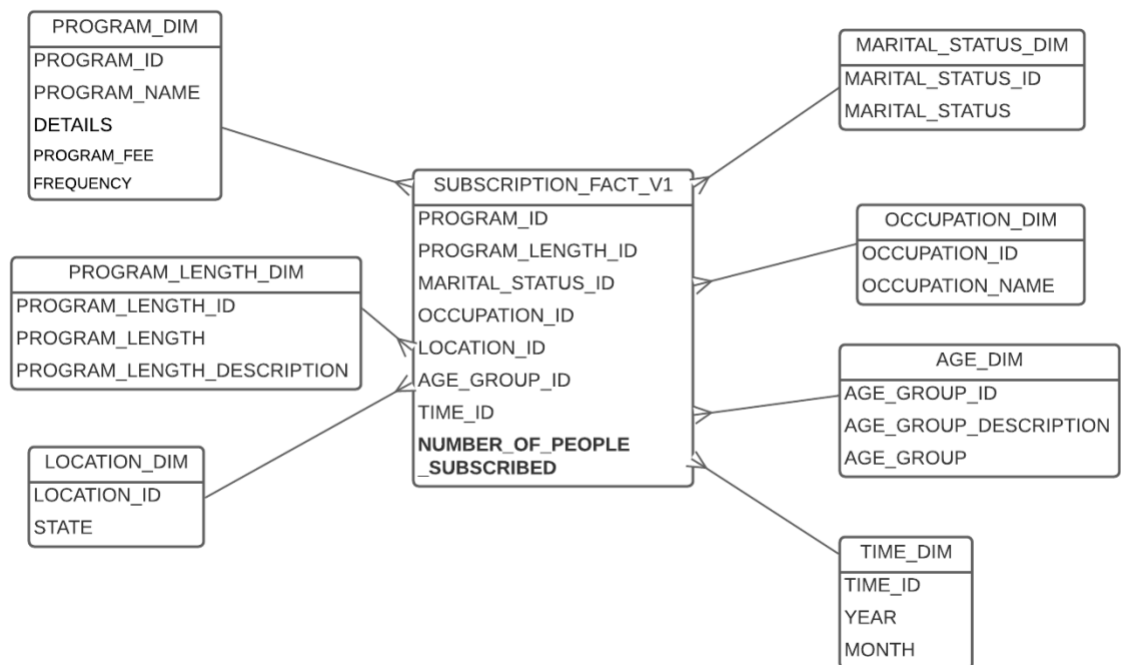
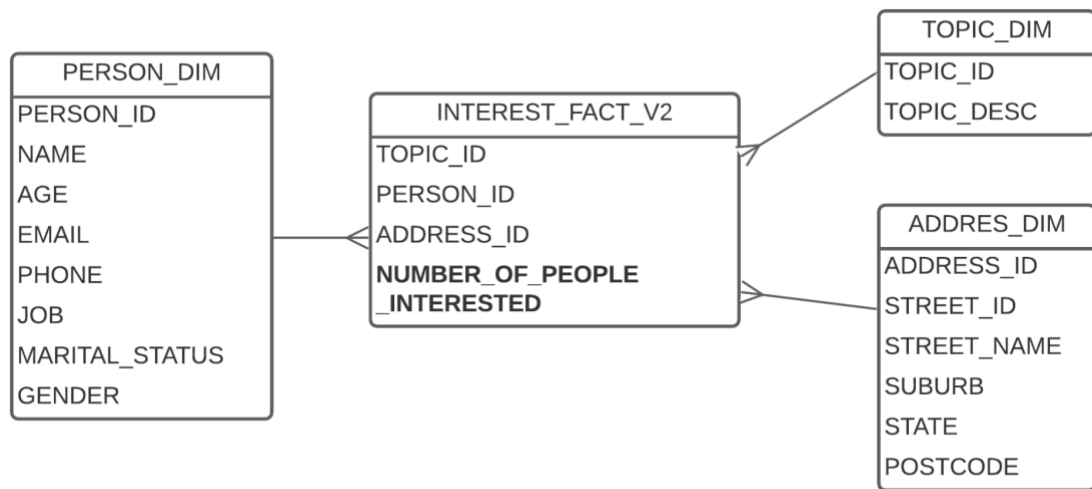
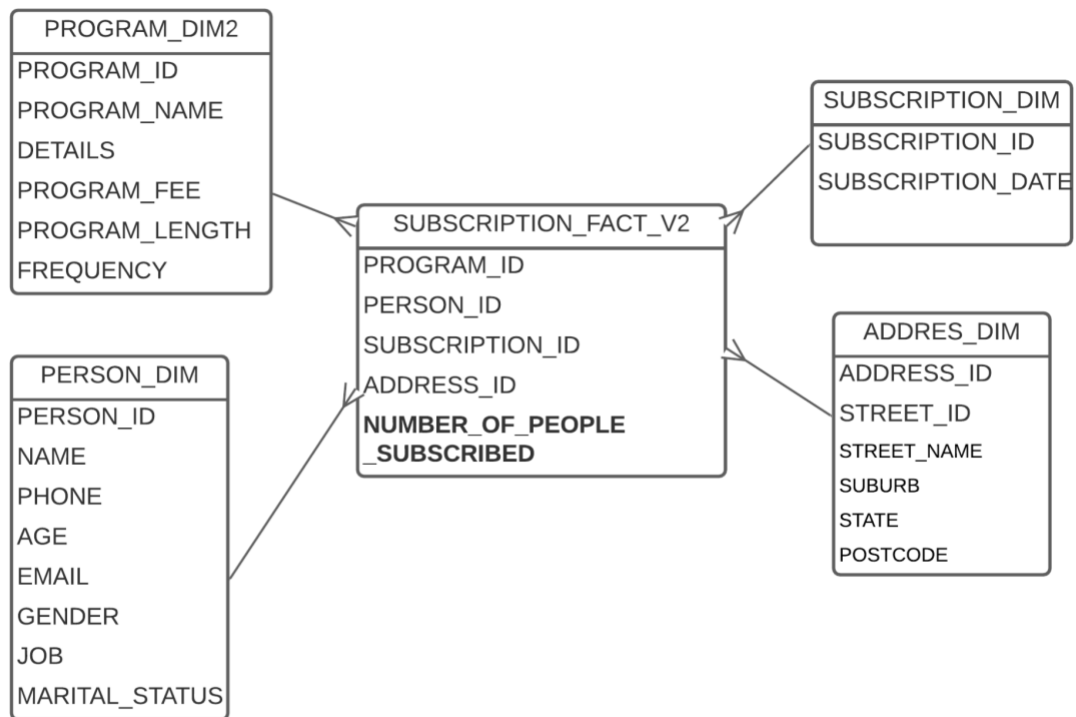


Figure 65 : SUBSCRIPTION_FACT_V1

LEVEL 0

**Figure 66** INTEREST_FACT_V2**Figure 67** SUBSCRIPTION_FACT_V2

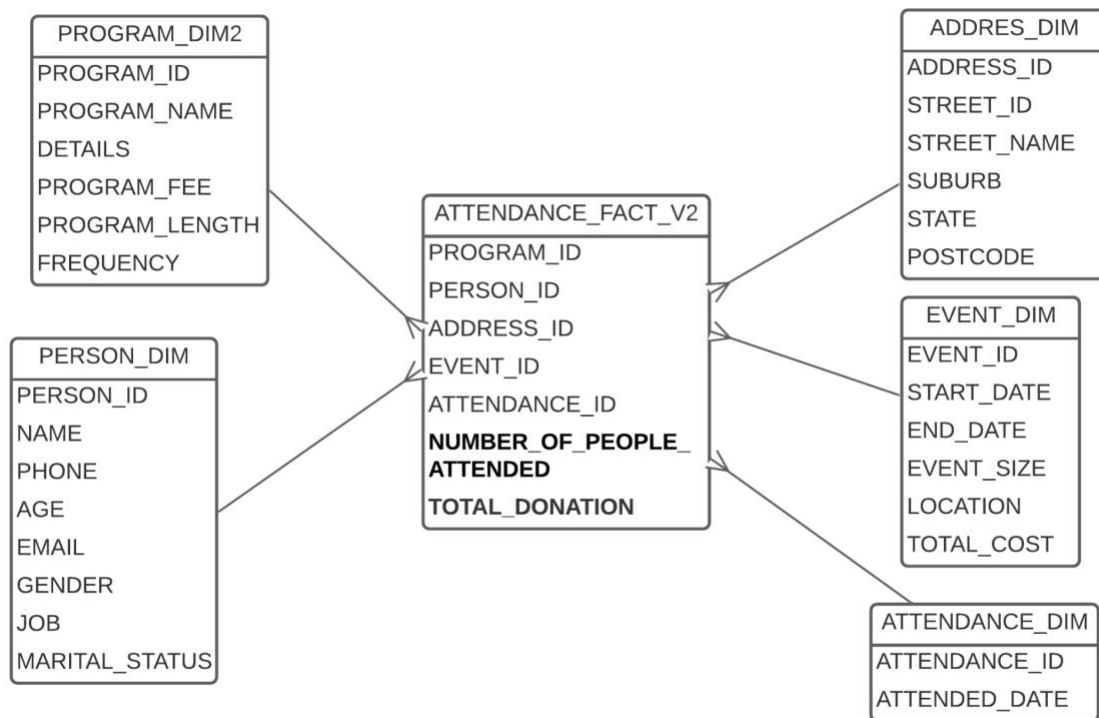


Figure 68 ATTENDANCE_FACT_V2

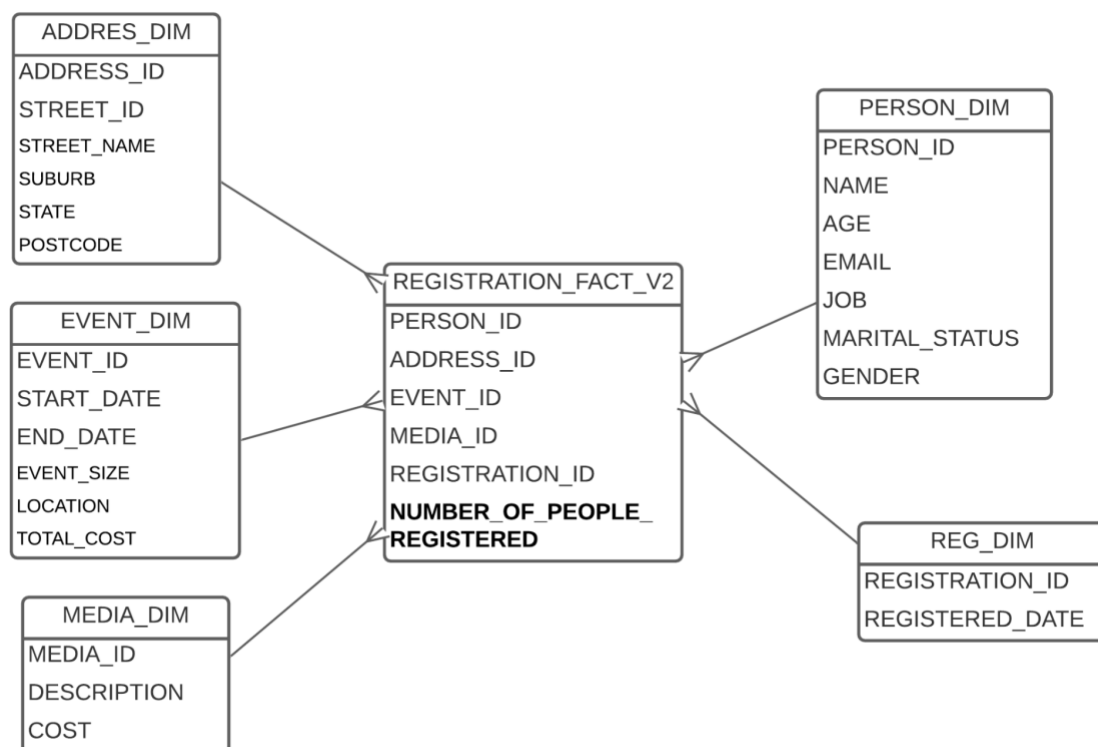


Figure 69 REGISTRATION_FACT_V2

d) The reasons for the choice of SCD type for the temporal dimension

Due to the historical changes of event costs over time, where events are run for a particular program, we decided to come up with the Slowly Changing Dimensions (SCD) for the program - PROGRAM_EVENT_HISTORY. As SCD type, SCD type 4 was chosen, based on the data stored in the database for the events, it provides management with the opportunity to drill down the program and see the full history of events related to this program with information about dates, costs and event location. Such information will help management with the analysis of the programs and related events. Another reason to choose SCD type 4 is that we do not need to provide information about current and previous costs, and current flag, mostly because the events happened in the past, and there is no such term as current cost for them, it is a cost for that particular event. Having SCD type 4 we have the same identifier for program_id, we don't need to change it, hence it is a good choice, which satisfies management needs and provides them with tools for analysis. Having looked for event location, dates, and costs, management can decide about the further events and run them and forecast the costs.

e) A short explanation of the difference among the two versions of star/snowflake schema.

Level 2 star schema is more general in form of aggregation. The facts created in level 2 does grouping to give aggregation. For example, the dimensions in level 2 have group-level attributes like marital_status, occupation and age_group. These attributes try to group a person from the given operational database based on different attributes. Another example of grouping is for the program, which has dimension like program_length. This dimension creates a group of programs based on the sessions. Similarly, for Event, event_size groups the events as Small, Medium and large events. Also, in level 2, the time_dim discussed has a granularity of month and year level. This changes in Level 0 to the actual date. The location_dim in level 2, considers granularity at the level of the state. Again this changes in Level 0 to include more details from the operational database.

For Level 0, the data becomes more narrow and close to the operational database. The level of details increases as we move to level 0, for example, in level 0 we include all the attributes of the Person for the person_dim. This thus removes the need to have group-level dimensions like age_group, occupation, marital_status, etc. In the case of program and Event, we copy the whole table as a dimension. The address_dim now has more granular data, capturing the street_id, street_name, suburb, state and postcode. This level of detail is the difference between level 0 and level 2 schemas.

To have a more clear view of the difference between level 0 and level 2, let's look at the number of people in attended fact. Let's run the following 2 queries and look at the number of rows returned.

```
SELECT count(*) COUNT_ROWS_V2 FROM ATTENDANCE_FACT_V2 WHERE
PROGRAM_ID = 'PR014';
```

	COUNT_ROWS_V2
1	600

Figure 70 Select query results

```
SELECT count(*) COUNT_ROWS_V1 FROM ATTENDANCE_FACT_V1 WHERE
PROGRAM_ID = 'PR014';
```

	COUNT_ROWS_V1
1	35

Figure 71 Select query results

We can see that the number of rows for level 0 is 600 which is more than the number of rows returned for level 2. This is because of the granularity of data in both the case. As discussed, for level 0 the data is more granular and thus has more records for the query, as compared to level 2 which groups the data and thus decreases the row count.

C. 2 IMPLEMENTATION

VERSION-1 STAR SCHEMA - HIGH AGGREGATION (LEVEL 2)

CREATING DIMENSION TABLES.

TOPIC_DIM:

```
CREATE TABLE TOPIC_DIM AS
SELECT *
FROM TOPIC;
Table TOPIC_DIM created. |
```

Figure 72 Create table message

Check the content of created table:

	TOPIC_ID	TOPIC_DESCRIPTION
1	T001	Networking
2	T002	Health and Lifestyle
3	T003	Spirituality
4	T004	Art and Culture
5	T005	Sport and Hobbies

Figure 73 Content of TOPIC_DIM table

LOCATION_DIM

The reason to use the only state in the location dimension based on the questions interesting to management and on the level of aggregation 2. To increase granularity it is

possible to include suburbs and postcodes, but we believe that those attributes are more suitable for level 1, which is not part of the assignment.

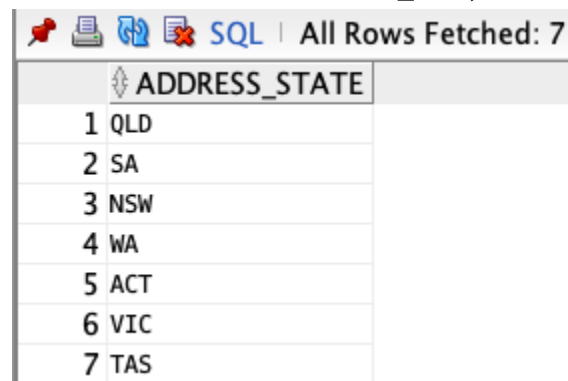
```
CREATE TABLE LOCATION_DIM AS
SELECT DISTINCT ADDRESS_STATE
FROM ADDRESS;
```

Table LOCATION_DIM created.

Figure 74 Create table message

Check the content of a created table

```
SELECT * FROM LOCATION_DIM;
```



ADDRESS_STATE
1 QLD
2 SA
3 NSW
4 WA
5 ACT
6 VIC
7 TAS

Figure 75 Content of LOCATION_DIM

Alter table to add Location ID

```
ALTER TABLE LOCATION_DIM ADD (LOCATION_ID NUMBER(1));
```

Table LOCATION_DIM altered.

Figure 76 Alter table message

Create a sequence to populate Location ID

```
CREATE SEQUENCE LOCATION_SEQ_ID
START WITH 1
INCREMENT BY 1
MAXVALUE 10
MINVALUE 1
NOCYCLE;
```

Sequence LOCATION_SEQ_ID created.

Figure 77 Sequence create message

Update LOCATION_DIM with LOCATION_ID values

```
UPDATE LOCATION_DIM SET LOCATION_ID = LOCATION_SEQ_ID.NEXTVAL;
```

7 rows updated.

Figure 78 update table message

Drop sequence

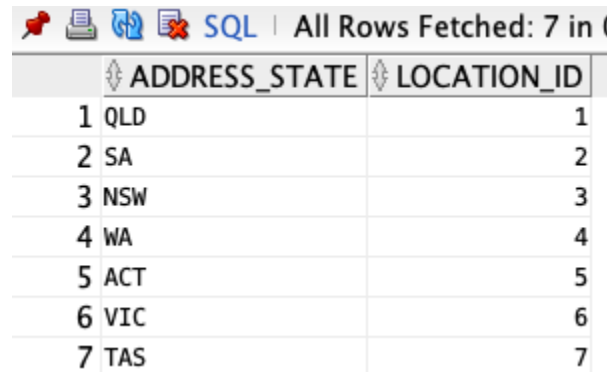
```
DROP SEQUENCE LOCATION_SEQ_ID;
```

Sequence LOCATION_SEQ_ID dropped.

Figure 79 drop sequence message

Check the content of the amended table

```
SELECT * FROM LOCATION_DIM;
```



	ADDRESS_STATE	LOCATION_ID
1	QLD	1
2	SA	2
3	NSW	3
4	WA	4
5	ACT	5
6	VIC	6
7	TAS	7

Figure 80 Content of LOCATION_DIM

MARITAL_STATUS_DIM

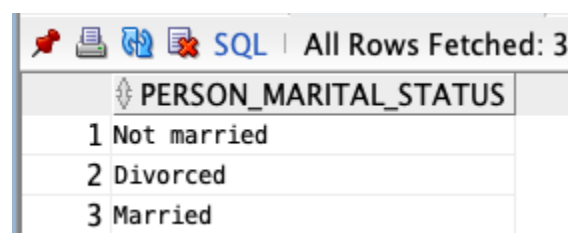
```
CREATE TABLE MARITAL_STATUS_DIM AS
SELECT DISTINCT PERSON_MARITAL_STATUS
FROM PERSON;
```

Table MARITAL_STATUS_DIM created.

Figure 81 Create table message

Check the content of a created table

```
SELECT * FROM MARITAL_STATUS_DIM;
```



	PERSON_MARITAL_STATUS
1	Not married
2	Divorced
3	Married

Figure 82 Content of MARITAL_STATUS_DIM

Alter table to add Marital_satus_id

```
ALTER TABLE MARITAL_STATUS_DIM ADD (MARITAL_STATUS_ID NUMBER(1));
```

Table MARITAL_STATUS_DIM altered.

Figure 83 alter the table message

Create a sequence to populate Location ID

```
CREATE SEQUENCE MARITAL_STATUS_SEQ_ID
START WITH 1
```

```

INCREMENT BY 1
MAXVALUE 10
MINVALUE 1
NOCYCLE;

```

Sequence MARITAL_STATUS_SEQ_ID created.

Figure 84 create a sequence message

```

Update LOCATION_DIM with LOCATION_ID values
UPDATE MARITAL_STATUS_DIM SET MARITAL_STATUS_ID =
MARITAL_STATUS_SEQ_ID.NEXTVAL;
3 rows updated.

```

Figure 85 update table message

```

Drop sequence
DROP SEQUENCE MARITAL_STATUS_SEQ_ID;

Sequence MARITAL_STATUS_SEQ_ID dropped.





```

Figure XX drop sequence message

```

Check the content of an amended table
SELECT * FROM MARITAL_STATUS_DIM;

```

    SQL | All Rows Fetched: 3 in 0.012 seconds

	PERSON_MARITAL_STATUS	MARITAL_STATUS_ID
1	Not married	1
2	Divorced	2
3	Married	3

Figure 86 Content of amended MARITAL_STATUS_DIM

OCCUPATION_DIM

```

CREATE TABLE OCCUPATION_DIM
(
OCCUPATION_ID NUMBER(1),
OCCUPATION_NAME VARCHAR2(20)
);

```

Table OCCUPATION_DIM created.

Figure 87 Table creation message

```

Insert values into the table
INSERT INTO OCCUPATION_DIM VALUES (1, 'Student');
INSERT INTO OCCUPATION_DIM VALUES (2, 'Staff');
INSERT INTO OCCUPATION_DIM VALUES (3, 'Community');

```

1 row inserted.

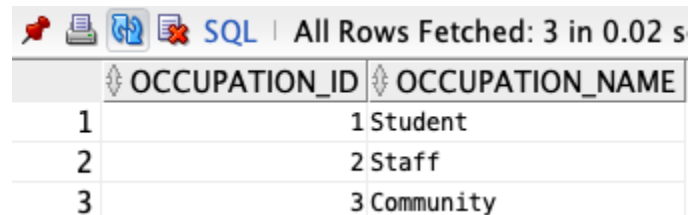
1 row inserted.

1 row inserted.

Figure 88 insert row message

Checking the content of a created table

SELECT * FROM OCCUPATION_DIM;



	OCCUPATION_ID	OCCUPATION_NAME
1	1	Student
2	2	Staff
3	3	Community

Figure 89 content of OCCUPATION_DIM

AGE_DIM

CREATE TABLE AGE_DIM

```
(
  AGE_GROUP_ID NUMBER(1),
  AGE_GROUP VARCHAR2(20),
  AGE_GROUP_DESCRIPTION VARCHAR2(20)
);
```

Table AGE_DIM created.

Figure 90 Create table message

Insert values into a table

INSERT INTO AGE_DIM VALUES (1, 'Child', '0-16 years old');

INSERT INTO AGE_DIM VALUES (2, 'Young adults', '17-30 years old');

INSERT INTO AGE_DIM VALUES (3, 'Middle-aged adults', '31-45 years old');

INSERT INTO AGE_DIM VALUES (4, 'Old-aged adults', 'Over 45 years old');

1 row inserted.

1 row inserted.

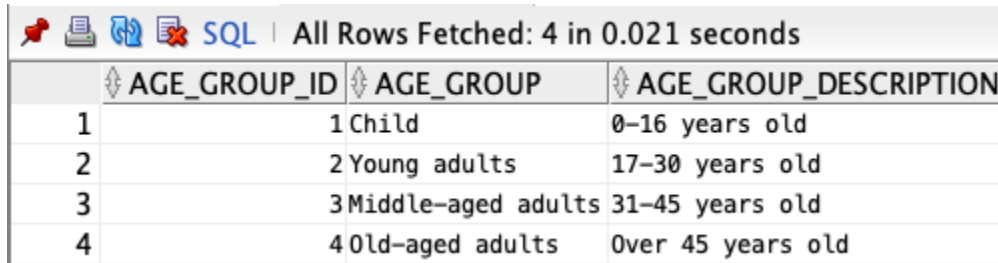
1 row inserted.

1 row inserted.

Figure 91 insert values into the table message

Checking the content of a created table

SELECT * FROM AGE_DIM;



	AGE_GROUP_ID	AGE_GROUP	AGE_GROUP_DESCRIPTION
1	1	Child	0-16 years old
2	2	Young adults	17-30 years old
3	3	Middle-aged adults	31-45 years old
4	4	Old-aged adults	Over 45 years old

Figure 92 Content of AGE_DIM

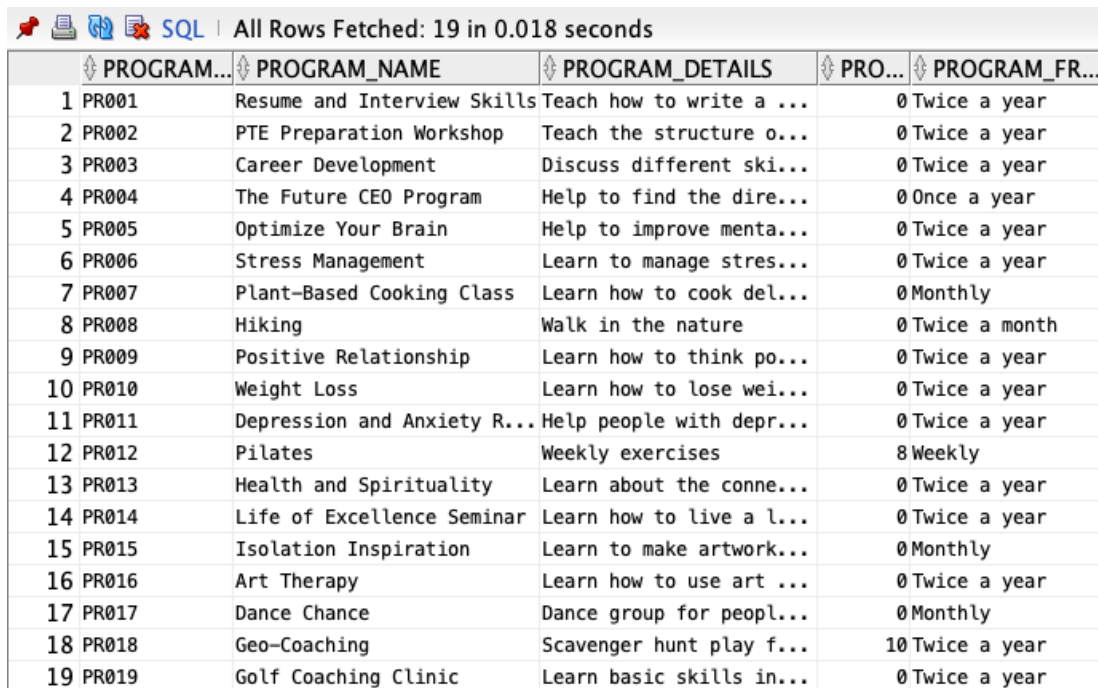
PROGRAM_DIM

```
CREATE TABLE PROGRAM_DIM AS
SELECT PROGRAM_ID, PROGRAM_NAME, PROGRAM_DETAILS, PROGRAM_FEE,
PROGRAM_FREQUENCY
FROM PROGRAM;
```

Table PROGRAM_DIM created.

Figure 93 Create table message

Check the content of a created table



	PROGRAM_ID	PROGRAM_NAME	PROGRAM_DETAILS	PROGRAM_FEE	PROGRAM_FREQUENCY
1	PR001	Resume and Interview Skills	Teach how to write a ...	0	Twice a year
2	PR002	PTE Preparation Workshop	Teach the structure o...	0	Twice a year
3	PR003	Career Development	Discuss different ski...	0	Twice a year
4	PR004	The Future CEO Program	Help to find the dire...	0	Once a year
5	PR005	Optimize Your Brain	Help to improve menta...	0	Twice a year
6	PR006	Stress Management	Learn to manage stres...	0	Twice a year
7	PR007	Plant-Based Cooking Class	Learn how to cook del...	0	Monthly
8	PR008	Hiking	Walk in the nature	0	Twice a month
9	PR009	Positive Relationship	Learn how to think po...	0	Twice a year
10	PR010	Weight Loss	Learn how to lose wei...	0	Twice a year
11	PR011	Depression and Anxiety R...	Help people with depr...	0	Twice a year
12	PR012	Pilates	Weekly exercises	8	Weekly
13	PR013	Health and Spirituality	Learn about the conne...	0	Twice a year
14	PR014	Life of Excellence Seminar	Learn how to live a l...	0	Twice a year
15	PR015	Isolation Inspiration	Learn to make artwork...	0	Monthly
16	PR016	Art Therapy	Learn how to use art ...	0	Twice a year
17	PR017	Dance Chance	Dance group for peopl...	0	Monthly
18	PR018	Geo-Coaching	Scavenger hunt play f...	10	Twice a year
19	PR019	Golf Coaching Clinic	Learn basic skills in...	0	Twice a year

Figure 94 Content of PROGRAM_DIM table

PROGRAM_LENGTH_DIM

```
CREATE TABLE PROGRAM_LENGTH_DIM
(
PROGRAM_LENGTH_ID NUMBER(1),
```

```
PROGRAM_LENGTH VARCHAR2(20),
PROGRAM_LENGTH_DESCRIPTION VARCHAR2(40)
);
```

Table PROGRAM_LENGTH_DIM created.

Figure 95 table creation message

Insert values into the table

```
INSERT INTO PROGRAM_LENGTH_DIM VALUES (1, 'short', 'event < three sessions');
INSERT INTO PROGRAM_LENGTH_DIM VALUES (2, 'medium', 'event between three to
six sessions');
INSERT INTO PROGRAM_LENGTH_DIM VALUES (3, 'long', 'event > six sessions');
1 row inserted.
```

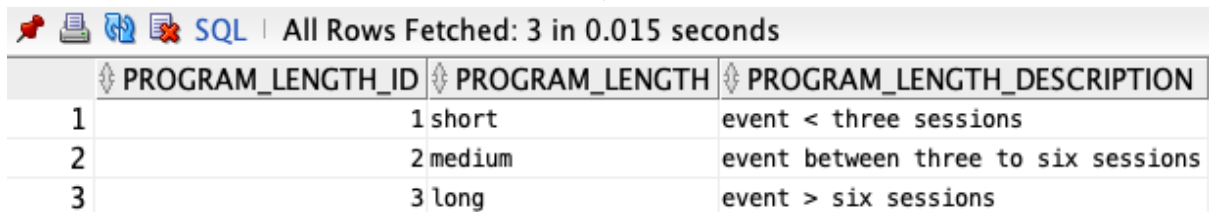
1 row inserted.

1 row inserted.

Figure 96 rows insert message

Checking the content of a created table

```
SELECT * FROM PROGRAM_LENGTH_DIM;
```



	PROGRAM_LENGTH_ID	PROGRAM_LENGTH	PROGRAM_LENGTH_DESCRIPTION
1	1	short	event < three sessions
2	2	medium	event between three to six sessions
3	3	long	event > six sessions

Figure 97 content of PROGRAM_LENGTH_DIM

TIME_DIM

```
CREATE TABLE TIME_DIM AS
SELECT
DISTINCT TO_CHAR(SUBSCRIPTION_DATE, 'YYYYMM') AS TIME_ID,
TO_CHAR(SUBSCRIPTION_DATE, 'YYYY') AS YEAR,
TO_CHAR(SUBSCRIPTION_DATE, 'MM') AS MONTH
FROM SUBSCRIPTION
UNION
SELECT
DISTINCT TO_CHAR(ATT_DATE, 'YYYYMM') AS TIME_ID,
TO_CHAR(ATT_DATE, 'YYYY') AS YEAR,
TO_CHAR(ATT_DATE, 'MM') AS MONTH
FROM ATTENDANCE
UNION
SELECT
DISTINCT TO_CHAR(REG_DATE, 'YYYYMM') AS TIME_ID,
TO_CHAR(REG_DATE, 'YYYY') AS YEAR,
```

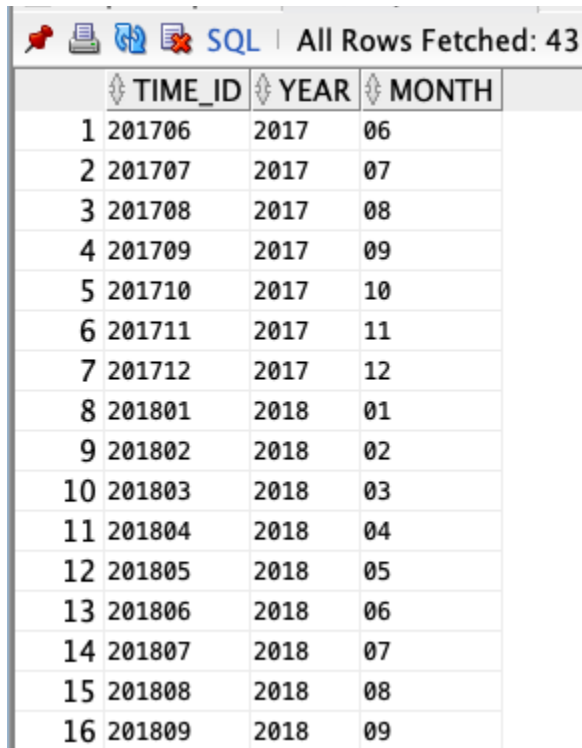
```
TO_CHAR(REG_DATE, 'MM') AS MONTH
FROM REGISTRATION;
```

Table TIME_DIM created.

Figure 98 table creation message

Checking the content of a created table

```
SELECT * FROM TIME_DIM;
```



	TIME_ID	YEAR	MONTH
1	201706	2017	06
2	201707	2017	07
3	201708	2017	08
4	201709	2017	09
5	201710	2017	10
6	201711	2017	11
7	201712	2017	12
8	201801	2018	01
9	201802	2018	02
10	201803	2018	03
11	201804	2018	04
12	201805	2018	05
13	201806	2018	06
14	201807	2018	07
15	201808	2018	08
16	201809	2018	09

Figure 99 Content of TIME_DIM (shown partly)

Dimension TIME_DIM created from a union of 3 tables - Subscription, Attendance and Registration because these tables contain date information required for further analysis by management.

PROGRAM_EVENT_HISTORY_DIM

```
CREATE TABLE PROGRAM_EVENT_HISTORY_DIM
AS SELECT DISTINCT PROGRAM_ID, EVENT_START_DATE, EVENT_END_DATE,
EVENT_COST
FROM EVENT;
```

Table PROGRAM_EVENT_HISTORY_DIM created.

Figure 100 Create table message

```
SELECT * FROM PROGRAM_EVENT_HISTORY_DIM;
```

	PROGRAM_ID	EVENT_START_DATE	EVENT_END_DATE	EVENT_COST	EVENT_LOCATION
1	PR012	10/APR/18	10/APR/18	10	MonExplore
2	PR018	30/MAY/18	02/JUN/18	10	MonExplore
3	PR012	09/JUL/18	09/JUL/18	40	MonExplore
4	PR007	28/SEP/18	28/SEP/18	30	Online
5	PR016	29/OCT/18	29/OCT/18	0	Online
6	PR017	23/NOV/18	23/NOV/18	20	Online
7	PR007	27/DEC/18	27/DEC/18	0	Online
8	PR002	15/JAN/19	17/JAN/19	40	Online
9	PR007	26/JAN/19	26/JAN/19	10	Online
10	PR016	27/APR/19	27/APR/19	40	MonExplore
11	PR007	25/JUN/19	25/JUN/19	0	MonExplore
12	PR005	16/AUG/19	11/OCT/19	40	Online
13	(null)	02/SEP/19	02/SEP/19	30	MonExplore
14	PR007	23/SEP/19	23/SEP/19	0	Online

Figure 101 Content of PROGRAM_EVENT_HISTORY_DIM

EVENT_SIZE_DIM

```
CREATE TABLE EVENT_SIZE_DIM
(
  EVENT_SIZE_ID NUMBER(1),
  EVENT_SIZE VARCHAR2(20),
  EVENT_SIZE_DESCRIPTION VARCHAR2(40)
);
Table EVENT_SIZE_DIM created.
```

Figure 102 create a table message

Insert values into the table

```
INSERT INTO EVENT_SIZE_DIM VALUES (1, 'small', 'event <= 10 people');
INSERT INTO EVENT_SIZE_DIM VALUES (2, 'medium', 'event between 11 and 30
people');
INSERT INTO EVENT_SIZE_DIM VALUES (3, 'large', 'event > 30 people');
1 row inserted.

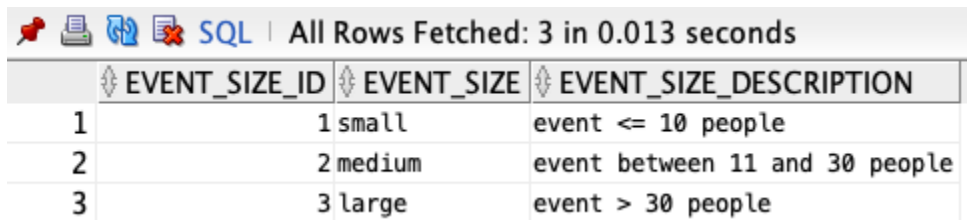
1 row inserted.

1 row inserted.
```

Figure 103 rows insert message

Checking the content of a created table

```
SELECT * FROM EVENT_SIZE_DIM;
```



SQL | All Rows Fetched: 3 in 0.013 seconds

	EVENT_SIZE_ID	EVENT_SIZE	EVENT_SIZE_DESCRIPTION
1	1	small	event <= 10 people
2	2	medium	event between 11 and 30 people
3	3	large	event > 30 people

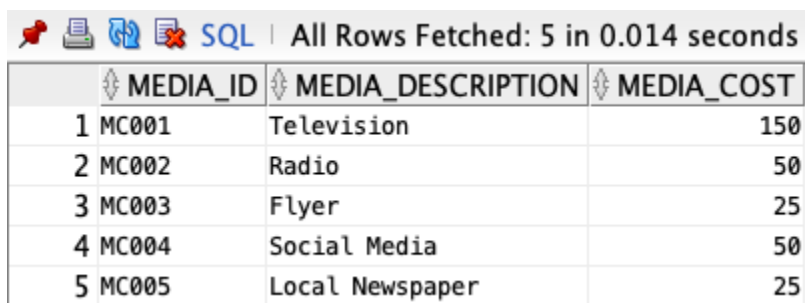
Figure 104 Content of EVENT_SIZE_DIM

MEDIA_DIM

```
CREATE TABLE MEDIA_DIM AS
SELECT *
FROM MEDIA_CHANNEL;
Table MEDIA_DIM created.
```

Figure 105 create a table message

Checking the content of a the created table
 SELECT * FROM MEDIA_DIM;



SQL | All Rows Fetched: 5 in 0.014 seconds

	MEDIA_ID	MEDIA_DESCRIPTION	MEDIA_COST
1	MC001	Television	150
2	MC002	Radio	50
3	MC003	Flyer	25
4	MC004	Social Media	50
5	MC005	Local Newspaper	25

Figure 106 Content of MEDIA_DIM

CREATING A FACT TABLES

INTEREST_FACT_V1

```
CREATE TABLE TEMP_INTEREST_FACT_V1 AS
SELECT T.TOPIC_ID, PE.PERSON_MARITAL_STATUS, (CASE
  WHEN PE.PERSON_JOB = 'Student' THEN 'Student'
  WHEN PE.PERSON_JOB = 'Staff' THEN 'Staff'
  ELSE 'Community'
END) AS OCCUPATION, A.ADDRESS_STATE,
(CASE
  WHEN PE.PERSON_AGE >=0 AND PE.PERSON_AGE <=16 THEN 'Child'
  WHEN PE.PERSON_AGE >=17 AND PE.PERSON_AGE <=30 THEN 'Young adults'
  WHEN PE.PERSON_AGE >=31 AND PE.PERSON_AGE <=45 THEN 'Middle-aged adults'
  ELSE 'Old-aged adults'
END) AS AGE
```

```
FROM TOPIC T, PERSON PE, ADDRESS A, PERSON_INTEREST PI
WHERE A.ADDRESS_ID = PE.ADDRESS_ID AND
PE.PERSON_ID = PI.PERSON_ID AND
PI.TOPIC_ID = T.TOPIC_ID;
```

Table TEMP_INTEREST_FACT_V1 created.

Figure 107 Create table message

Check content of a created table

```
SELECT * FROM TEMP_INTEREST_FACT_V1;
```



TOPIC_ID	PERSON_ID	OCCUPATION_ID	ADDRESS_ID	AGE
----------	-----------	---------------	------------	-----

Figure 108 Content of TEMP_INTEREST_FACT_V1

The table is empty because table PERSON_INTEREST is empty in the operational database after data cleaning.

Alter temp fact, add required attributes

```
ALTER TABLE TEMP_INTEREST_FACT_V1
ADD (MARITAL_STATUS_ID NUMBER(1));
```

```
ALTER TABLE TEMP_INTEREST_FACT_V1
ADD (OCCUPATION_ID NUMBER(1));
```

```
ALTER TABLE TEMP_INTEREST_FACT_V1
ADD (LOCATION_ID NUMBER(1));
```

```
ALTER TABLE TEMP_INTEREST_FACT_V1
ADD (AGE_GROUP_ID NUMBER(1));
```

Table TEMP_INTEREST_FACT_V1 altered.

Table TEMP_INTEREST_FACT_V1 altered.

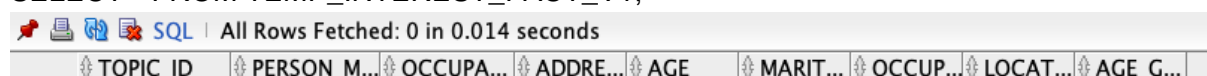
Table TEMP_INTEREST_FACT_V1 altered.

Table TEMP_INTEREST_FACT_V1 altered.

Figure 109 Alter table messages

Check altered table

```
SELECT * FROM TEMP_INTEREST_FACT_V1;
```



TOPIC_ID	PERSON_ID	OCCUPATION_ID	ADDRESS_ID	AGE	MARITAL_STATUS_ID	OCCUPATION_ID	LOCATION_ID	AGE_GROUP_ID
----------	-----------	---------------	------------	-----	-------------------	---------------	-------------	--------------

Figure 110 Content of altered table

The table is empty, but we can see newly the added attributes

Update values in the temp fact

```
UPDATE TEMP_INTEREST_FACT_V1 TF
SET TF.MARITAL_STATUS_ID = (SELECT S.MARITAL_STATUS_ID
FROM MARITAL_STATUS_DIM S
WHERE S.PERSON_MARITAL_STATUS = TF.PERSON_MARITAL_STATUS);
```

```
UPDATE TEMP_INTEREST_FACT_V1 TF
SET TF.OCCUPATION_ID = (SELECT S.OCCUPATION_ID
FROM OCCUPATION_DIM S
WHERE S.OCCUPATION_NAME = TF.OCCUPATION);
```

```
UPDATE TEMP_INTEREST_FACT_V1 TF
SET TF.LOCATION_ID = (SELECT S.LOCATION_ID
FROM LOCATION_DIM S
WHERE S.ADDRESS_STATE = TF.ADDRESS_STATE);
```

```
UPDATE TEMP_INTEREST_FACT_V1 TF
SET TF.AGE_GROUP_ID = (SELECT S.AGE_GROUP_ID
FROM AGE_DIM S
WHERE S.AGE_GROUP = TF.AGE);
```

```
0 rows updated. |
```

```
0 rows updated.
```

```
0 rows updated.
```

```
0 rows updated.
```

Figure 111 Update table messages

Zero rows updated because our table is empty.

Create INTEREST_FACT_V1

```
CREATE TABLE INTEREST_FACT_V1 AS
SELECT TOPIC_ID, MARITAL_STATUS_ID, OCCUPATION_ID,
LOCATION_ID, AGE_GROUP_ID,
COUNT(*) AS NUMBEROFPEOPLEINTERESTED
FROM TEMP_INTEREST_FACT_V1
GROUP BY TOPIC_ID, MARITAL_STATUS_ID, OCCUPATION_ID,
LOCATION_ID, AGE_GROUP_ID;
```

```
Table INTEREST_FACT_V1 created.
```

Figure 112 Create table message

Check the content of a created table

```
SELECT * FROM INTEREST_FACT_V1;
```

SQL All Rows Fetched: 0 in 0.014 seconds						
TOPIC_ID	MARITAL_STATUS_ID	OCCUPATION_ID	LOCATION_ID	AGE_GROUP_ID	NUMBEROFPEO...	

Figure 113 Content of INTEREST_FACT_V1

Our fact table is empty due to the absence of data in the PERSON_INTEREST table.

Drop temp fact

```
DROP TABLE TEMP_INTEREST_FACT_V1;
```

Table TEMP_INTEREST_FACT_V1 dropped.

Figure 114 drop table message

SUBSCRIPTION_FACT_V1

Creating temp fact

```
CREATE TABLE TEMP_SUBSCRIPTION_FACT_V1 AS
SELECT P.PROGRAM_ID,P.PROGRAM_LENGTH,
PE.PERSON_MARITAL_STATUS,PE.PERSON_JOB , a.address_state,
PE.PERSON_AGE, S.SUBSCRIPTION_DATE
FROM PROGRAM P, PERSON PE, ADDRESS A, SUBSCRIPTION S
WHERE A.ADDRESS_ID = PE.ADDRESS_ID AND
PE.PERSON_ID = S.PERSON_ID AND
S.PROGRAM_ID = P.PROGRAM_ID;
```

Table TEMP_SUBSCRIPTION_FACT_V1 created.

Figure 115 Create table message

Checking the content of a created table

```
SELECT * FROM TEMP_SUBSCRIPTION_FACT_V1;
```

SQL Fetched 100 rows in 0.026 seconds						
PROGRAM_ID	PROGRAM_LENGTH	PERSON_MARITAL_STATUS	PERSON_JOB	ADDRESS_STATE	PERSON_AGE	SUBSCRIPTION_DATE
1 PR002	2 sessions	Not married	Accountants	QLD	55 18.07.17	
2 PR008	1 session	Not married	Staff	WA	41 01.07.17	
3 PR007	1 session	Not married	Student	WA	27 05.12.17	
4 PR013	12 sessions	Married	Student	NSW	27 18.10.17	
5 PR009	1 session	Married	Staff	VIC	37 10.08.17	
6 PR002	2 sessions	Not married	Staff	QLD	38 03.06.17	
7 PR001	1 session	Not married	Staff	NSW	42 21.08.17	
8 PR019	1 session	Not married	Insurance Sales...	SA	60 06.11.17	
9 PR016	1 session	Married	Laborers and Fr...	TAS	57 15.09.17	
10 PR015	14 sessions	Married	Accountants	WA	52 22.11.17	
11 PR016	1 session	Not married	Student	ACT	33 13.09.17	
12 PR008	1 session	Married	Student	VIC	28 22.12.17	

Figure 116 content of TEMP_SUBSCRIPTION_FACT_V1 (shown partly)

Altering temp fact, adding required attributes:

add PROGRAM_LENGTH_ID

```
ALTER TABLE TEMP_SUBSCRIPTION_FACT_V1
```

```
ADD (PROGRAM_LENGTH_ID NUMBER(1));
```

add MARITAL_STATUS_ID


```
ALTER TABLE TEMP_SUBSCRIPTION_FACT_V1
ADD (MARITAL_STATUS_ID NUMBER(1));
```

```
add OCCUPATION_ID
ALTER TABLE TEMP_SUBSCRIPTION_FACT_V1
ADD (OCCUPATION_ID NUMBER(1));
```

```
add LOCATION_ID
ALTER TABLE TEMP_SUBSCRIPTION_FACT_V1
ADD (LOCATION_ID NUMBER(1));
```

```
add AGE_GROUP_ID
ALTER TABLE TEMP_SUBSCRIPTION_FACT_V1
ADD (AGE_GROUP_ID NUMBER(1));
```

```
add TIME_ID
ALTER TABLE TEMP_SUBSCRIPTION_FACT_V1
ADD (TIME_ID VARCHAR(6));
```

```
add PROGRAM_LENGTH_SIZE
ALTER TABLE TEMP_SUBSCRIPTION_FACT_V1
ADD (PROGRAM_LENGTH_SIZE NUMBER(3));
```

```
Table TEMP_SUBSCRIPTION_FACT_V1 altered.
```

```
Table TEMP_SUBSCRIPTION_FACT_V1 altered.
```

```
Table TEMP_SUBSCRIPTION_FACT_V1 altered.
```

```
Table TEMP_SUBSCRIPTION_FACT_V1 altered.
```

```
Table TEMP_SUBSCRIPTION_FACT_V1 altered.
```

```
Table TEMP_SUBSCRIPTION_FACT_V1 altered.
```

```
Table TEMP_SUBSCRIPTION_FACT_V1 altered.
```

Figure 117 Alter table messages

Updating values of created attributes

Update PROGRAM_LENGTH_SIZE

```
UPDATE TEMP_SUBSCRIPTION_FACT_V1
```

```
SET PROGRAM_LENGTH_SIZE = SUBSTR(PROGRAM_LENGTH, 1,
INSTR(PROGRAM_LENGTH, '-')-1);
```

Update PROGRAM_LENGTH_ID

```

UPDATE TEMP_SUBSCRIPTION_FACT_V1
SET PROGRAM_LENGTH_ID = CASE
WHEN PROGRAM_LENGTH_SIZE < 3 THEN 1
WHEN PROGRAM_LENGTH_SIZE >=3 AND PROGRAM_LENGTH_SIZE <=6 THEN 2
ELSE 3
END;

```

```

Update MARITAL_STATUS_ID
UPDATE TEMP_SUBSCRIPTION_FACT_V1 TF
SET TF.MARITAL_STATUS_ID = (SELECT S.MARITAL_STATUS_ID
FROM MARITAL_STATUS_DIM S
WHERE S.PERSON_MARITAL_STATUS = TF.PERSON_MARITAL_STATUS);

```

```

Update OCCUPATION_ID
UPDATE TEMP_SUBSCRIPTION_FACT_V1
SET OCCUPATION_ID = CASE
WHEN PERSON_JOB = 'Student' THEN 1
WHEN PERSON_JOB = 'Staff' THEN 2
ELSE 3
END;

```

```

Update LOCATION_ID
UPDATE TEMP_SUBSCRIPTION_FACT_V1 TF
SET TF.LOCATION_ID = (SELECT S.LOCATION_ID
FROM LOCATION_DIM S
WHERE S.ADDRESS_STATE = TF.ADDRESS_STATE);

```

```

Update AGE_GROUP_ID
UPDATE TEMP_SUBSCRIPTION_FACT_V1 TF
SET TF.AGE_GROUP_ID = CASE
WHEN PERSON_AGE >=0 AND PERSON_AGE <=16 THEN 1
WHEN PERSON_AGE >=17 AND PERSON_AGE <=30 THEN 2
WHEN PERSON_AGE >=31 AND PERSON_AGE <=45 THEN 3
ELSE 4
END;

```

```

Update TIME_ID
UPDATE TEMP_SUBSCRIPTION_FACT_V1
SET TIME_ID = TO_CHAR(SUBSCRIPTION_DATE,'YYYYMM');

```

500 rows updated.

500 rows updated.

500 rows updated.

500 rows updated.

500 rows updated.

500 rows updated.

500 rows updated.

Figure 118 update rows messages

Checking the content of an altered table

SELECT * FROM TEMP_SUBSCRIPTION_FACT_V1;

SQL | Fetched 50 rows in 0.013 seconds

LENGTH	PERSON_MARITAL_STATUS	PERSON_JOB	ADDRE...	PERSON...	SUBSC...	PR...	MARIT...	OCCUP...	LOCAT...	AGE_G...	TIME_ID	PROGR...
2	Married	Student	NSW	27 18.10.17	3	3	3	1	3	2 201710	12	
3	Married	Staff	VIC	37 10.08.17	1	3	2	6	3 201708	1		
4	Not married	Staff	QLD	38 03.06.17	1	1	2	1	3 201706	2		
5	Not married	Staff	NSW	42 21.08.17	1	1	2	3	3 201708	1		
6	Not married	Insurance Sal...	SA	60 06.11.17	1	1	3	2	4 201711	1		
7	Married	Laborers and ...	TAS	57 15.09.17	1	3	3	7	4 201709	1		
8	Married	Accountants	WA	52 22.11.17	3	3	3	4	4 201711	14		
9	Not married	Student	ACT	33 13.09.17	1	1	1	5	3 201709	1		
10	Married	Student	NSW	27 01.12.17	1	3	1	3	2 201712	1		
11	Not married	Student	NSW	29 05.07.17	2	1	1	3	2 201707	3		
12	Married	Student	SA	32 08.10.17	3	3	1	2	3 201710	8		
13	Married	Student	TAS	29 10.12.17	2	3	1	7	2 201712	3		
14	Married	Staff	VIC	36 13.11.17	3	3	2	6	3 201711	12		
15	Not married	Staff	NSW	39 04.08.17	3	1	2	3	3 201708	12		
16	Not married	Staff	WA	42 03.09.17	3	1	2	4	3 201709	8		
17	Divorced	Student	QLD	26 18.08.17	1	2	1	1	2 201708	1		
18	Not married	Network and C...	NSW	50 13.09.17	1	1	3	3	4 201709	1		

Figure 119 Check content of TEMP_SUBSCRIPTION_FACT_V1

Commit changes

COMMIT;

Commit complete.

Figure 120 commit message

Create SUBSCRIPTION_FACT_V1

```
CREATE TABLE SUBSCRIPTION_FACT_V1 AS
SELECT PROGRAM_ID, PROGRAM_LENGTH_ID, MARITAL_STATUS_ID,
OCCUPATION_ID, LOCATION_ID, AGE_GROUP_ID, TIME_ID, COUNT(*) AS
NUMBEROFPEOPLESUBSCRIBED
FROM TEMP_SUBSCRIPTION_FACT_V1
GROUP BY PROGRAM_ID, PROGRAM_LENGTH_ID, MARITAL_STATUS_ID,
OCCUPATION_ID, LOCATION_ID, AGE_GROUP_ID, TIME_ID;
```

Table SUBSCRIPTION_FACT_V1 created.

Figure 121 Create table message

Checking the content of a created table

SELECT * FROM SUBSCRIPTION_FACT_V1;

SQL | Fetched 200 rows in 0.052 seconds

	PROG...	PR...	MARITAL_...	OCC...	LOCAT...	AGE...	TIME_ID	NUMBEROFPEOPLESUBSCRIBED
7	PR013	3	1	2	2	3	201708	1
8	PR006	3	1	2	4	3	201706	1
9	PR002	1	1	3	4	4	201708	1
10	PR002	1	3	3	7	4	201707	1
11	PR009	1	1	3	3	4	201707	1
12	PR010	3	2	2	4	3	201706	2
13	PR016	1	3	1	6	3	201707	1
14	PR019	1	1	3	1	4	201710	1
15	PR009	1	1	3	1	4	201710	1
16	PR004	3	3	1	1	3	201706	1
17	PR011	3	2	2	2	3	201712	1
18	PR016	1	1	3	3	4	201707	1
19	PR010	3	1	3	3	4	201706	1
20	PR008	1	1	2	1	3	201712	1
21	PR013	3	3	2	6	3	201708	1
22	PR015	3	1	1	1	3	201711	1
23	PR005	3	2	1	1	3	201706	2
24	PR008	1	3	2	1	3	201707	1

Figure 122 Content of SUBSCRIPTION_FACT_V1 (shown partly)

Drop temp fact

DROP TABLE TEMP_SUBSCRIPTION_FACT_V1;

Table TEMP_SUBSCRIPTION_FACT_V1 dropped.

Figure 123 drop table message

REGISTRATION_FACT_V1

Create temp fact

```
CREATE table TEMP_REGISTRATION_FACT_V1 AS
SELECT (CASE
WHEN PE.PERSON_MARITAL_STATUS = 'Not married' THEN 1
WHEN PE.PERSON_MARITAL_STATUS = 'Divorced' THEN 2
ELSE 3
END) AS MARITAL_STATUS_ID, (CASE
WHEN PE.PERSON_JOB = 'Student' then 1
when PE.PERSON_JOB = 'Staff' then 2
else 3
end) AS OCCUPATION_ID, a.address_state, (CASE
WHEN PE.person_age >=0 and PE.person_age <=16 then 1
when PE.person_age >=17 and PE.person_age <=30 then 2
when PE.person_age >=31 and PE.person_age <=45 then 3
```

```

else 4
end) AS AGE_GROUP_ID, (CASE
WHEN E.EVENT_SIZE<= 10 THEN 1
when E.EVENT_SIZE >10 and E.EVENT_SIZE <=30 then 2
else 3
end) AS EVENT_SIZE_ID, M.MEDIA_ID, TO_CHAR(R.REG_DATE, 'YYYYMM') AS
TIME_ID, R.REG_NUM_OF_PEOPLE_REGISTERED
FROM ADDRESS A, PERSON PE, REGISTRATION R, MEDIA_CHANNEL M, EVENT E
WHERE A.ADDRESS_ID = PE.ADDRESS_ID AND
PE.PERSON_ID = R.PERSON_ID AND
R.EVENT_ID = E.EVENT_ID AND
R.MEDIA_ID = M.MEDIA_ID;


































```

Table TEMP_REGISTRATION_FACT_V1 created.

Figure 124 Create table message

Check content of created table

```
select * from TEMP_REGISTRATION_FACT_V1;
```





                                

```
UPDATE TEMP_REGISTRATION_FACT_V1 TF
SET TF.LOCATION_ID = (SELECT S.LOCATION_ID
FROM LOCATION_DIM S
WHERE S.ADDRESS_STATE = TF.ADDRESS_STATE);
```

1,492 rows updated.

Figure 127 rows update message

Check the content of an altered and updated table
select * from TEMP_REGISTRATION_FACT_V1;

    SQL | Fetched 50 rows in 0.015 seconds

	MAR...	OCCU...	ADD...	AGE...	EVEN...	MEDIA_ID	TIME_ID	REG_NUM_OF_...	LOCAT...
1	1	3 SA		4	3 MC003	201803		2	2
2	3	3 NSW		4	3 MC001	201801		2	3
3	1	2 NSW		3	3 MC002	201908		4	3
4	3	3 TAS		4	3 MC004	201812		2	7
5	1	2 VIC		3	3 MC005	201806		4	6
6	3	1 VIC		3	3 MC001	202002		4	6
7	1	2 NSW		3	3 MC005	201808		1	3
8	1	1 WA		2	3 MC005	202006		1	4
9	1	3 SA		4	3 MC003	202002		3	2
10	1	1 VIC		2	3 MC004	201903		2	6
11	2	3 ACT		4	3 MC005	201801		4	5
12	3	3 WA		4	2 MC005	201812		1	4
13	1	2 QLD		3	3 MC002	202008		4	1
14	1	1 WA		2	3 MC001	201903		1	4
15	1	1 VIC		2	3 MC005	201903		1	6
16	1	2 NSW		3	2 MC002	202005		1	3
17	1	3 NSW		4	3 MC001	202008		4	3
18	2	2 WA		3	3 MC004	201908		3	4

Figure 128 Content of TEMP_REGISTRATION_FACT_V1 (shown partly)

Create **REGISTRATION_FACT_V1**

```
CREATE TABLE REGISTRATION_FACT_V1 AS
SELECT MARITAL_STATUS_ID, OCCUPATION_ID, LOCATION_ID, AGE_GROUP_ID,
EVENT_SIZE_ID, MEDIA_ID, TIME_ID, SUM(REG_NUM_OF_PEOPLE_REGISTERED)
AS NumberOfPeopleRegistered
FROM TEMP_REGISTRATION_FACT_V1
GROUP BY MARITAL_STATUS_ID, OCCUPATION_ID, LOCATION_ID, AGE_GROUP_ID,
EVENT_SIZE_ID, MEDIA_ID, TIME_ID ;
```

Table REGISTRATION_FACT_V1 created.

Figure 129 Create table message

Check the content of a created table

```
SELECT * FROM REGISTRATION_FACT_V1;
```

SQL | Fetched 50 rows in 0.023 seconds

	MARITAL_STATUS_ID	OCCUPATION_ID	LOCATION_ID	AGE_GROUP_ID	EVENT_SIZE_ID	MEDIA_ID	TIME_ID	NUMBEROFPEOPLEREGISTERED
1	1	1	6	2	3 MC005	201903		1
2	2	2	2	3	3 MC002	201907		1
3	3	3	7	4	3 MC003	201907		1
4	1	1	4	2	3 MC005	201811		2
5	3	1	6	2	3 MC003	201805		4
6	3	2	1	3	2 MC001	201802		2
7	1	2	1	3	3 MC004	201806		1
8	3	1	3	2	3 MC005	201905		1
9	3	2	1	3	3 MC001	201802		4
10	1	1	1	3	2 MC001	201908		2
11	1	2	4	3	3 MC004	201902		1
12	3	1	3	2	3 MC003	202007		4
13	1	1	3	2	3 MC005	202002		7
14	3	3	1	4	3 MC004	201807		4
15	1	2	1	3	3 MC003	201801		2
16	1	3	3	4	2 MC004	201908		2
17	1	2	3	3	3 MC001	201809		4
18	3	1	6	2	3 MC001	201804		2

Figure 130 Content of REGISTRATION_FACT_V1

Drop temp fact

```
DROP TABLE TEMP_REGISTRATION_FACT_V1;
```

Table TEMP_REGISTRATION_FACT_V1 dropped.

Figure 131 Drop table message

ATTENDANCE_FACT_V1

```
CREATE TABLE ATTENDANCE_TEMP_FACT
AS
SELECT
    PR.PROGRAM_ID,
    PR.PROGRAM_LENGTH,
    PE.PERSON_MARITAL_STATUS,
    PE.PERSON_JOB,
    AD.ADDRESS_STATE,
    PE.PERSON_AGE,
    ev.event_size,
    att.att_date,
    ATT.ATT_NUM_OF_PEOPLE_ATTENDED,
    ATT.ATT_DONATION_AMOUNT
FROM
    PROGRAM PR JOIN EVENT EV
        ON PR.PROGRAM_ID = EV.PROGRAM_ID
    JOIN ATTENDANCE ATT
        ON ATT.EVENT_ID = EV.EVENT_ID
    JOIN PERSON PE
        ON PE.PERSON_ID = ATT.PERSON_ID
    JOIN ADDRESS AD
        ON AD.ADDRESS_ID = PE.ADDRESS_ID;
```

```
Table ATTENDANCE_TEMP_FACT created.
```

Figure 132 Create table message

```
ALTER TABLE ATTENDANCE_TEMP_FACT ADD (PROGRAM_LENGTH_ID NUMBER(2));
ALTER TABLE ATTENDANCE_TEMP_FACT ADD (MARITAL_STATUS_ID NUMBER(2));
ALTER TABLE ATTENDANCE_TEMP_FACT ADD (OCCUPATION_ID NUMBER(2));
ALTER TABLE ATTENDANCE_TEMP_FACT ADD (LOCATION_ID NUMBER(2));
ALTER TABLE ATTENDANCE_TEMP_FACT ADD (AGE_GROUP_ID NUMBER(2));
ALTER TABLE ATTENDANCE_TEMP_FACT ADD (EVENT_SIZE_ID NUMBER(2));
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

Figure 133 Alter table messages

```
ALTER TABLE ATTENDANCE_TEMP_FACT
ADD (PROGRAM_LENGTH_SIZE NUMBER(3));
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

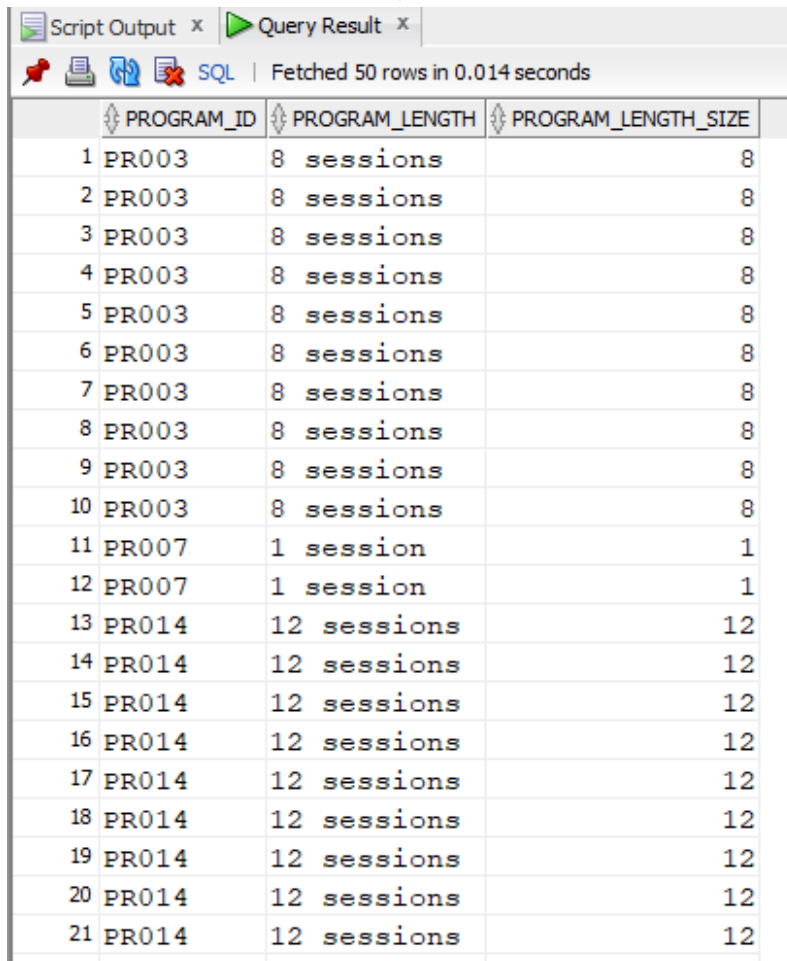
Figure 134 Alter table message

```
UPDATE ATTENDANCE_TEMP_FACT ATT_TF
SET ATT_TF.PROGRAM_LENGTH_SIZE = SUBSTR(ATT_TF.PROGRAM_LENGTH, 1,
INSTR(ATT_TF.PROGRAM_LENGTH, '-')-1);
```


5,751 rows updated.

Figure 135 Rows update message

```
SELECT PROGRAM_ID, PROGRAM_LENGTH, PROGRAM_LENGTH_SIZE FROM
ATTENDANCE_TEMP_FACT;
```



	PROGRAM_ID	PROGRAM_LENGTH	PROGRAM_LENGTH_SIZE
1	PR003	8 sessions	8
2	PR003	8 sessions	8
3	PR003	8 sessions	8
4	PR003	8 sessions	8
5	PR003	8 sessions	8
6	PR003	8 sessions	8
7	PR003	8 sessions	8
8	PR003	8 sessions	8
9	PR003	8 sessions	8
10	PR003	8 sessions	8
11	PR007	1 session	1
12	PR007	1 session	1
13	PR014	12 sessions	12
14	PR014	12 sessions	12
15	PR014	12 sessions	12
16	PR014	12 sessions	12
17	PR014	12 sessions	12
18	PR014	12 sessions	12
19	PR014	12 sessions	12
20	PR014	12 sessions	12
21	PR014	12 sessions	12

Figure 136 Content of the ATTENDANCE_TEMP_FACT

```
UPDATE ATTENDANCE_TEMP_FACT
SET PROGRAM_LENGTH_ID = 1
WHERE PROGRAM_LENGTH_SIZE < 3;
```

1,052 rows updated.

Figure 137 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT
SET PROGRAM_LENGTH_ID = 2
```

```
WHERE PROGRAM_LENGTH_SIZE >= 3  
AND PROGRAM_LENGTH_SIZE <= 6 ;
```

```
147 rows updated.
```

Figure 138 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET PROGRAM_LENGTH_ID = 2  
WHERE PROGRAM_LENGTH_SIZE > 6 ;
```

```
4,552 rows updated.
```

Figure 139 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET MARITAL_STATUS_ID = 1  
WHERE PERSON_MARITAL_STATUS = 'Not married' ;
```

```
2,527 rows updated.
```

Figure 140 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET MARITAL_STATUS_ID = 1  
WHERE PERSON_MARITAL_STATUS = 'Divorced' ;
```

```
1,136 rows updated.
```

Figure 141 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET MARITAL_STATUS_ID = 1  
WHERE PERSON_MARITAL_STATUS = 'Married' ;
```

```
2,088 rows updated.
```

Figure 142 Rows update message

```
select PERSON_MARITAL_STATUS, MARITAL_STATUS_ID FROM  
ATTENDANCE_TEMP_FACT;
```

PERSON_MARITAL_STATUS	MARITAL_STATUS_ID
1 Married	3
2 Married	3
3 Divorced	2
4 Divorced	2
5 Divorced	2
6 Divorced	2
7 Divorced	2
8 Divorced	2
9 Married	3
10 Divorced	2
11 Not married	1
12 Not married	1

Figure 143 Content of the ATTENDANCE_TEMP_FACT

```
UPDATE ATTENDANCE_TEMP_FACT
SET OCCUPATION_ID = 1
WHERE person_job = 'Student' ;
```

```
2,030 rows updated.
```

Figure 144 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT
SET OCCUPATION_ID = 2
WHERE person_job = 'Staff' ;
```

```
1,958 rows updated.
```


Figure 145 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT
SET OCCUPATION_ID = 3
WHERE person_job NOT IN ('Student','Staff') ;
```

```
1,763 rows updated.
```

Figure 146 Rows update message


```
UPDATE ATTENDANCE_TEMP_FACT
SET LOCATION_ID = 1
WHERE ADDRESS_STATE = 'QLD' ;
```



```
1,256 rows updated.
```

Figure 147 Rows update message


```
UPDATE ATTENDANCE_TEMP_FACT  
SET LOCATION_ID = 2  
WHERE ADDRESS_STATE = 'SA' ;
```



```
792 rows updated.
```

Figure 148 Rows update message


```
UPDATE ATTENDANCE_TEMP_FACT  
SET LOCATION_ID = 3  
WHERE ADDRESS_STATE = 'NSW' ;
```



```
1,330 rows updated.
```

Figure 149 Rows update message


```
UPDATE ATTENDANCE_TEMP_FACT  
SET LOCATION_ID = 4  
WHERE ADDRESS_STATE = 'WA' ;
```



```
1,074 rows updated.
```

Figure 150 Rows update message


```
UPDATE ATTENDANCE_TEMP_FACT  
SET LOCATION_ID = 5  
WHERE ADDRESS_STATE = 'ACT' ;
```



```
106 rows updated.
```

Figure 151 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET LOCATION_ID = 6  
WHERE ADDRESS_STATE = 'VIC' ;
```



```
987 rows updated.
```

Figure 152 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET LOCATION_ID = 7  
WHERE ADDRESS_STATE = 'TAS' ;
```

```
206 rows updated.
```

Figure 153 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET AGE_GROUP_ID = 1  
WHERE PERSON_AGE <= 16;
```

```
0 rows updated.
```

Figure 154 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET AGE_GROUP_ID = 2  
WHERE PERSON_AGE >= 17  
AND PERSON_AGE <=30;
```

```
946 rows updated.
```

Figure 155 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET AGE_GROUP_ID = 3  
WHERE PERSON_AGE >= 31  
AND PERSON_AGE <=45;
```

```
3,042 rows updated.
```

Figure 156 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET AGE_GROUP_ID = 4  
WHERE PERSON_AGE > 45;
```

```
1,763 rows updated.
```

Figure 157 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET EVENT_SIZE_ID = 1  
WHERE EVENT_SIZE <= 10;
```

```
149 rows updated.
```

Figure 158 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET EVENT_SIZE_ID = 2  
WHERE EVENT_SIZE >= 11  
AND EVENT_SIZE <= 30;
```

```
948 rows updated.
```

Figure 159 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT  
SET EVENT_SIZE_ID = 3  
WHERE EVENT_SIZE > 30;
```

```
4,654 rows updated.
```

Figure 160 Rows update message

```
ALTER TABLE ATTENDANCE_TEMP_FACT ADD (TIME_ID VARCHAR2(6));
```

```
Table ATTENDANCE_TEMP_FACT altered.
```

Figure 161 Rows update message

```
UPDATE ATTENDANCE_TEMP_FACT ATT  
SET TIME_ID = TO_CHAR(ATT.ATT_DATE, 'YYYYMM');
```

```
5,751 rows updated.
```

Figure 162 Rows update message

ATTENDANCE FACT TABLE

```
CREATE TABLE ATTENDANCE_FACT_V1
```

```

AS
SELECT
  PROGRAM_ID,
  PROGRAM_LENGTH_ID,
  MARITAL_STATUS_ID,
  OCCUPATION_ID,
  LOCATION_ID,
  AGE_GROUP_ID,
  EVENT_SIZE_ID,
  SUM(ATT_NUM_OF_PEOPLE_ATTENDED) NO_PEOPLE_ATTENDED,
  SUM(ATT_DONATION_AMOUNT) TOTAL_DONATION
FROM
  ATTENDANCE_TEMP_FACT
GROUP BY
  PROGRAM_ID,
  PROGRAM_LENGTH_ID,
  MARITAL_STATUS_ID,
  OCCUPATION_ID,
  LOCATION_ID,
  AGE_GROUP_ID,
  EVENT_SIZE_ID;

```

	PROGRAM_ID	PROGRAM_LENGTH_ID	MARITAL_STATUS_ID	OCCUPATION_ID	LOCATION_ID	AGE_GROUP_ID	EVENT_SIZE_ID	NO_PEOPLE_ATTENDED	TOTAL_DONATION
1	PR001	1	3	3	3	4	3	9	60
2	PR012	1	1	2	6	3	3	23	85
3	PR005	2	3	2	4	3	3	78	735
4	PR005	2	3	3	1	4	3	43	530
5	PR008	1	2	3	4	4	3	13	165
6	PR011	2	3	1	6	2	3	64	465
7	PR015	2	2	1	1	3	2	87	595
8	PR015	2	2	2	2	3	3	69	600
9	PR007	1	1	2	1	3	3	35	370
10	PR003	2	3	1	6	3	3	147	1180
11	PR012	1	3	1	3	3	3	17	155
12	PR012	1	1	1	3	2	3	27	430
13	PR011	2	1	3	3	4	2	148	1385
14	PR005	2	1	2	2	3	3	396	3115
15	PR010	2	3	3	7	4	2	34	335
16	PR012	1	1	1	5	3	3	9	90
17	PR011	2	3	1	1	3	2	37	445
18	PR008	1	1	2	3	3	3	13	115
19	PR007	1	3	1	7	2	3	27	190
20	PR001	1	1	3	2	4	3	13	150
21	PR003	2	3	1	3	3	2	44	395
22	PR012	1	3	2	6	3	2	13	170
23	PR005	2	1	2	4	3	3	132	1195

Figure 163 Content of the ATTENDANCE_FACT_V1

VERSION-2 NO AGGREGATION (LEVEL 0)

CREATING DIMENSIONS:

EVENT_DIM





```
CREATE TABLE EVENT_DIM AS
SELECT EVENT_ID, EVENT_START_DATE, EVENT_END_DATE, EVENT_SIZE,
EVENT_LOCATION, EVENT_COST
FROM EVENT;
```

Table EVENT_DIM created.

Figure 164 create a table message

Check the content of the created table

```
SELECT * FROM EVENT_DIM;
```

    | All Rows Fetched: 170 in 0.086 seconds

	EVENT_ID	EVENT_START_DATE	EVENT_END_DATE	EVENT_SIZE	EVENT_LOCATION	EVENT_COST
1	1	10.01.18	10.01.18	89	Online	20
2	2	15.01.18	15.01.18	95	Online	10
3	3	20.01.18	22.01.18	82	Online	0
4	4	23.01.18	20.03.18	99	MonExplore	10
5	5	31.01.18	31.01.18	91	MonExplore	0
6	6	09.02.18	09.02.18	45	MonExplore	20
7	7	22.02.18	19.12.18	85	MonExplore	10
8	8	22.02.18	19.04.18	64	Online	0
9	9	25.02.18	22.04.18	46	Online	20
10	10	28.02.18	28.02.18	55	Online	0
11	11	02.03.18	02.03.18	10	MonExplore	0
12	12	02.03.18	02.03.18	88	Online	40
13	13	05.03.18	12.03.18	30	MonExplore	30
14	14	08.03.18	03.05.18	22	MonExplore	40
15	15	10.03.18	02.06.18	61	MonExplore	40
16	16	11.03.18	11.03.18	34	Online	20
17	17	13.03.18	05.06.18	98	Online	40
18	18	01.04.18	01.04.18	67	Online	0
19	19	07.04.18	21.04.18	71	Online	20
20	20	10.04.18	10.04.18	80	MonExplore	10

Figure 165 content of EVENT_DIM

PROGRAM_DIM2

```
CREATE TABLE PROGRAM_DIM2 AS
SELECT PROGRAM_ID, PROGRAM_NAME, PROGRAM_DETAILS,
PROGRAM_FEE, PROGRAM_LENGTH, PROGRAM_FREQUENCY
FROM PROGRAM;
```

Table PROGRAM_DIM2 created.

Figure 166 table creation message

Check content of a created table

```
SELECT * FROM PROGRAM_DIM2;
```


SQL | All Rows Fetched: 19 in 0.025 seconds

PROGRAM_ID	PROGRAM_NAME	PROGR...	PROGRAM_FEE	PROGRAM_LENGTH	PROGRAM_FREQ
1 PR001	Resume and Interview Skills	Teach ho...	01 session	Twice a year	
2 PR002	PTE Preparation Workshop	Teach th...	02 sessions	Twice a year	
3 PR003	Career Development	Discuss ...	08 sessions	Twice a year	
4 PR004	The Future CEO Program	Help to ...	010 sessions	Once a year	
5 PR005	Optimize Your Brain	Help to ...	08 sessions	Twice a year	
6 PR006	Stress Management	Learn to...	08 sessions	Twice a year	
7 PR007	Plant-Based Cooking Class	Learn ho...	01 session	Monthly	
8 PR008	Hiking	Walk in ...	01 session	Twice a month	
9 PR009	Positive Relationship	Learn ho...	01 session	Twice a year	
10 PR010	Weight Loss	Learn ho...	07 sessions	Twice a year	
11 PR011	Depression and Anxiety Recovery	Help peo...	08 sessions	Twice a year	
12 PR012	Pilates	Weekly e...	81 session	Weekly	
13 PR013	Health and Spirituality	Learn ab...	012 sessions	Twice a year	
14 PR014	Life of Excellence Seminar	Learn ho...	012 sessions	Twice a year	
15 PR015	Isolation Inspiration	Learn to...	014 sessions	Monthly	
16 PR016	Art Therapy	Learn ho...	01 session	Twice a year	
17 PR017	Dance Chance	Dance gr...	01 session	Monthly	
18 PR018	Geo-Coaching	Scavenge...	103 sessions	Twice a year	
19 PR019	Golf Coaching Clinic	Learn ba...	01 session	Twice a year	

Figure 167 Content of PROGRAM_DIM2

The difference between PROGRAM_DIM and PROGRAM_DIM2 is there is a PROGRAM_LENGTH attribute included in PROGRAM_DIM2 due to a lower level of aggregation.

PERSON_DIM

```
CREATE TABLE PERSON_DIM AS
SELECT PERSON_ID, PERSON_NAME, PERSON_PHONE, PERSON_AGE,
PERSON_EMAIL, PERSON_GENDER, PERSON_JOB, PERSON_MARITAL_STATUS
FROM PERSON;
```

Table PERSON_DIM created.

Figure 168 Create table message

Check the content of a created table
 SELECT * FROM PERSON_DIM;

SQL | All Rows Fetched: 100 in 0.059 seconds

	PERSON_ID	PERSON_NAME	PERSON_PHONE	PERSON_AGE	PERSON_EMAIL	PERSON_GENDER
1	PE006	Patrick Crowe	0429251617	55	PatrickCrowe@teleworm.us	F
2	PE015	Tayla Fitz	0489053141	30	TaylaFitz@jourrapide.com	F
3	PE018	Poppy Trenerry	0474077071	54	PoppyTrenerry@rhyta.com	F
4	PE024	Beau Julia	0424953370	59	BeauJulia@teleworm.us	M
5	PE032	Jackson McConnan	0436293182	27	JacksonMcConnan@rhyta.com	F
6	PE043	Claire Downing	0424952943	41	ClaireDowning@einrot.com	M
7	PE052	Aiden Lyle	0424043918	36	AidenLyle@rhyta.com	F
8	PE062	Lily Cosh	0435365420	42	LilyCosh@rhyta.com	F
9	PE066	Eva Spode	0488967141	41	EvaSpode@fleckens.hu	M
10	PE067	Daniel Rouse	0488710896	36	DanielRouse@cuvox.de	F
11	PE072	Dominic Arthur	0474578194	36	DominicArthur@jourrapide.com	M
12	PE073	Bianca Parker	0475329919	59	BiancaParker@gustr.com	M
13	PE074	Dominic Stead	0474022260	52	DominicStead@cuvox.de	F
14	PE076	Edward Palmer	0424049409	39	EdwardPalmer@rhyta.com	M
15	PE079	Matilda Gunter	0474558280	34	MatildaGunter@fleckens.hu	F
16	PE096	David Newson	0435304337	32	DavidNewson@jourrapide.com	M
17	PE100	Cameron Nyhan	0436265432	34	CameronNyhan@jourrapide.com	F
18	PE004	Stephanie Seidel	0435322031	38	StephanieSeidel@armyspy.com	F
19	PE011	Charlotte Sugden	0474509194	45	CharlotteSugden@teleworm.us	F

Figure 169 Content of PERSON_DIM (shown partly)

SUBSCRIPTION_DIM

```
CREATE TABLE SUBSCRIPTION_DIM AS
SELECT SUBSCRIPTION_ID, SUBSCRIPTION_DATE
FROM SUBSCRIPTION;
```

Table SUBSCRIPTION_DIM created.

Figure 170 create a table message

Check the content of a created table
 SELECT * FROM SUBSCRIPTION_DIM;

SQL | All Rows Fetched: 500 in 0.13!

	SUBSCRIPTION_ID	SUBSCRIPTION_DATE
1	SU003	18.07.17
2	SU020	01.07.17
3	SU030	05.12.17
4	SU053	18.10.17
5	SU056	10.08.17
6	SU061	03.06.17
7	SU072	21.08.17
8	SU077	06.11.17
9	SU078	15.09.17
10	SU085	22.11.17
11	SU087	13.09.17
12	SU089	22.12.17
13	SU102	16.11.17
14	SU108	01.12.17
15	SU137	11.12.17
16	SU139	05.07.17
17	SU140	08.10.17
18	SU142	10.12.17
19	SU154	13.11.17
20	SU172	03.08.17

Figure 171 content of SUBSCRIPTION_DIM



ADDRES_DIM

```
CREATE TABLE ADDRES_DIM AS
SELECT *
FROM ADDRESS;
Table ADDRES_DIM created.
```

Figure 172 create a table message

Check the content of the the created table

```
SELECT * FROM ADDRES_DIM;
```

    SQL | All Rows Fetched: 100 in 0.044 seconds

ADDRESS_ID	ADDRESS_STREET_NO	ADDRESS_STREET_NAME	ADDRESS_SUBURB	ADDRESS_STATE	ADDRESS_POSTCODE
1 AD001	571	Walker Street	Upper Ferntree Gully	VIC	3156
2 AD002	851	Parsons Street	Murrays Run	NSW	2325
3 AD003	336	Barnett Street	Mount White	NSW	2250
4 AD004	914	Magistrates Walk	Veradilla	QLD	4347
5 AD005	799	Witchwood Close	Pimlico	QLD	4812
6 AD006	435	Rankins Road	Charters Towers City	QLD	4820
7 AD007	962	Macaulay Road	Borung	VIC	3518
8 AD008	743	Rankins Road	Trayning	WA	6488
9 AD009	859	Hotham Place	Fitzroy Crossing	WA	6765
10 AD010	908	Lambeth Street	Glenden	QLD	4743
11 AD011	774	Smith Street	Wheeny Creek	NSW	2758
12 AD012	262	Melrose Street	East Kempsey	NSW	2440
13 AD013	30	Lothian Street	Talisker	WA	6701
14 AD014	876	Epsom Road	Hope Vale	QLD	4895
15 AD015	999	Market Street	Beerwah	QLD	4519
16 AD016	476	Collett Street	Guildford	WA	6055
17 AD017	130	Magistrates Walk	Bourke	NSW	2840
18 AD018	777	Fisken Place	Lort River	WA	6447
19 AD019	562	Barnett Street	Mount Erin	WA	6532





Figure 173 content of ADDRES_DIM

TOPIC_DIM:

```
CREATE TABLE TOPIC_DIM AS
SELECT *
FROM TOPIC;
Table TOPIC_DIM created.
```

Figure 174 Create table message

Check the content of created table:

    SQL | All Rows Fetched: 5 in

TOPIC_ID	TOPIC_DESCRIPTION
1 T001	Networking
2 T002	Health and Lifestyle
3 T003	Spirituality
4 T004	Art and Culture
5 T005	Sport and Hobbies

Figure 175 Content of TOPIC_DIM

ATTENDANCE_DIM


```
CREATE TABLE ATTENDANCE_DIM AS
SELECT ATT_ID, ATT_DATE
FROM ATTENDANCE;
```

Table ATTENDANCE_DIM created.

Figure 176 Create a table message

Check the content of a created table

```
SELECT * FROM ATTENDANCE_DIM;
```

 SQL | Fetched 900 rows

	ATT_ID	ATT_DATE
1	620	04.09.19
2	621	11.09.19
3	622	25.01.19
4	623	01.02.19
5	624	08.02.19
6	625	15.02.19
7	626	22.02.19
8	627	01.03.19
9	628	08.03.19
10	629	15.03.19
11	630	27.03.20
12	631	30.07.18
13	632	24.09.20
14	633	01.10.20
15	634	08.10.20

Figure 177 Content of ATTENDANCE_DIM

MEDIA_DIM

```
CREATE TABLE MEDIA_DIM AS
SELECT *
FROM MEDIA_CHANNEL;
```

Table MEDIA_DIM created.

Figure 178 Create table message

Checking the content of a created table

```
SELECT * FROM MEDIA_DIM;
```

SQL | All Rows Fetched: 5 in 0.014 seconds

	MEDIA_ID	MEDIA_DESCRIPTION	MEDIA_COST
1	MC001	Television	150
2	MC002	Radio	50
3	MC003	Flyer	25
4	MC004	Social Media	50
5	MC005	Local Newspaper	25

Figure 179 Content of MEDIA_DIM

REG_DIM

```
CREATE TABLE REG_DIM AS
SELECT REG_ID, REG_DATE FROM REGISTRATION;
```

Table REG_DIM created.

Figure 180 Create table message

```
SELECT * FROM REG_DIM;
```

	REG_ID	REG_DATE
1	617	06/MAR/18
2	618	16/JAN/18
3	619	25/AUG/19
4	620	20/DEC/18
5	621	23/JUN/18
6	622	24/FEB/20
7	623	20/AUG/18
8	624	21/JUN/20
9	625	11/FEB/20
10	626	26/MAR/19
11	627	08/JAN/18
12	628	29/DEC/18

Figure 181 Content of REG_DIM

CREATING FACT TABLES

INTEREST_FACT_V2

```
CREATE TABLE INTEREST_FACT_V2
AS
SELECT
    T.TOPIC_ID,
    PE.PERSON_ID,
    AD.ADDRESS_ID,
```

```

COUNT(*) NO_PEOPLE_INTERESTED
FROM
  PERSON PE JOIN PERSON_INTEREST PI
    ON PE.PERSON_ID = PI.PERSON_ID
  JOIN TOPIC T
    ON T.TOPIC_ID = PI.TOPIC_ID
  JOIN ADDRESS AD
    ON AD.ADDRESS_ID = PE.ADDRESS_ID
GROUP BY
  T.TOPIC_ID, PE.PERSON_ID, AD.ADDRESS_ID;

```

Table INTEREST_FACT_V2 created.

Figure 182 Create table message

```
SELECT * FROM INTEREST_FACT_V2 ;
```

TOPIC_ID	PROGRA...	PERSON_ID	ADDRESS...	NO_PEOP...
----------	-----------	-----------	------------	------------

Figure 183 Content of INTEREST_FACT_V2

REGISTRATION_FACT_V2

```

CREATE TABLE REGISTRATION_FACT_V2
AS
  SELECT
    PE.PERSON_ID, AD.ADDRESS_ID,
    RE.EVENT_ID, RE.MEDIA_ID,
    RE.REG_ID,
    SUM(RE.REG_NUM_OF_PEOPLE_REGISTERED) NO_PEOPLE_REGISTERED
  FROM
    PERSON PE JOIN REGISTRATION RE
      ON PE.PERSON_ID = RE.PERSON_ID
    JOIN ADDRESS AD
      ON AD.ADDRESS_ID = PE.ADDRESS_ID
  GROUP BY
    PE.PERSON_ID, AD.ADDRESS_ID,
    RE.EVENT_ID, RE.MEDIA_ID,
    RE.REG_ID;

```

Table REGISTRATION_FACT_V2 created.

Figure 184 Create table message

```
SELECT * FROM REGISTRATION_FACT_V2;
```

	PERSON_ID	ADDRESS_ID	EVENT_ID	MEDIA_ID	REG_ID	NO_PEOPLE_REGISTERED
1	PE067	AD067	138	MC002	632	1
2	PE080	AD080	96	MC004	634	3
3	PE062	AD062	94	MC005	646	3
4	PE089	AD089	31	MC002	650	4
5	PE007	AD007	24	MC003	651	4
6	PE038	AD038	139	MC004	658	3
7	PE080	AD080	72	MC003	668	2
8	PE062	AD062	136	MC005	678	4
9	PE022	AD022	42	MC001	679	1
10	PE038	AD038	99	MC001	680	2
11	PE042	AD042	169	MC003	683	4
12	PE048	AD048	74	MC002	684	1
13	PE083	AD083	160	MC003	696	3
14	PE059	AD059	127	MC004	698	3
15	PE080	AD080	85	MC004	704	3
16	PE017	AD017	145	MC003	705	4

Figure 185 Content of REGISTRATION_FACT_V2

SUBSCRIPTION_FACT_V2

```
CREATE TABLE SUBSCRIPTION_FACT_V2
AS
SELECT
    PR.PROGRAM_ID,
    PE.PERSON_ID,
    SU.SUBSCRIPTION_ID,
    AD.ADDRESS_ID,
    COUNT(*) NO_PEOPLE_SUBSCRIPTION
FROM
    PERSON PE JOIN ADDRESS AD
    ON PE.ADDRESS_ID = AD.ADDRESS_ID
    JOIN SUBSCRIPTION SU
    ON SU.PERSON_ID = PE.PERSON_ID
    JOIN PROGRAM PR
    ON PR.PROGRAM_ID = SU.PROGRAM_ID
GROUP BY
    PR.PROGRAM_ID,
    PE.PERSON_ID,
    SU.SUBSCRIPTION_ID,
    AD.ADDRESS_ID;
```

Table SUBSCRIPTION_FACT_V2 created.

Figure 186 Create a table message

SELECT * FROM SUBSCRIPTION_FACT_V2;

	PROGRAM_ID	PERSON_ID	SUBSCRIPTION_ID	ADDRESS_ID	NO_PEOPLE_SUBSCRIPTION
1	PR019	PE056	SU077	AD056	1
2	PR016	PE022	SU087	AD022	1
3	PR018	PE093	SU139	AD093	1
4	PR004	PE085	SU409	AD085	1
5	PR019	PE060	SU450	AD060	1
6	PR012	PE083	SU475	AD083	1
7	PR009	PE028	SU073	AD028	1
8	PR003	PE082	SU118	AD082	1
9	PR010	PE073	SU122	AD073	1
10	PR018	PE034	SU164	AD034	1
11	PR007	PE068	SU168	AD068	1
12	PR002	PE073	SU374	AD073	1
13	PR002	PE080	SU397	AD080	1
14	PR011	PE019	SU150	AD019	1
15	PR017	PE035	SU209	AD035	1
16	PR012	PE074	SU241	AD074	1

Figure 187 Content of SUBSCRIPTION_FACT_V2

ATTENDANCE_FACT_V2

```
Create ATTENDANCE_FACT_V2
CREATE TABLE ATTENDANCE_FACT_V2 AS
SELECT P.PROGRAM_ID, PE.PERSON_ID, A.ADDRESS_ID, E.EVENT_ID,
ATT.ATT_ID, SUM(ATT.ATT_NUM_OF_PEOPLE_ATTENDED) AS
NumberOfPeopleAttended,
SUM(ATT.ATT_DONATION_AMOUNT) AS TotalDonation
FROM ADDRESS A, PERSON PE, ATTENDANCE ATT, EVENT E, PROGRAM P
WHERE A.ADDRESS_ID = PE.ADDRESS_ID AND
PE.PERSON_ID = ATT.PERSON_ID AND
ATT.EVENT_ID = E.EVENT_ID AND
E.PROGRAM_ID = P.PROGRAM_ID
GROUP BY P.PROGRAM_ID, PE.PERSON_ID, A.ADDRESS_ID, E.EVENT_ID,
ATT.ATT_ID;
```

Table ATTENDANCE_FACT_V2 created.

Figure 188 Create table message

Check the content of a created table

SELECT * FROM ATTENDANCE_FACT_V2;

SQL | Fetched 50 rows in 0.021 seconds

	PROGRAM_ID	PERSON_ID	ADDRESS_ID	EVENT_ID	ATT_ID	NUMBEROFPEOPLEATTENDED	TOTALDONATION
1	PR014	PE006	AD006	159	642	4	95
2	PR001	PE082	AD082	59	660	9	60
3	PR003	PE049	AD049	61	671	6	35
4	PR014	PE090	AD090	131	697	3	75
5	PR015	PE058	AD058	133	707	8	70
6	PR015	PE058	AD058	133	712	5	20
7	PR015	PE058	AD058	133	715	5	30
8	PR015	PE097	AD097	104	725	6	40
9	PR015	PE097	AD097	104	729	1	85
10	PR007	PE085	AD085	78	734	8	55
11	PR018	PE017	AD017	110	736	1	85
12	PR015	PE100	AD100	76	744	4	90
13	PR015	PE100	AD100	76	747	2	70
14	PR015	PE100	AD100	76	750	8	90
15	PR014	PE093	AD093	102	753	7	80
16	PR014	PE093	AD093	102	762	3	45
17	PR014	PE093	AD093	102	764	7	30
18	PR003	PE068	AD068	90	765	5	55

Figure 189Content of ATTENDANCE_FACT_V2 (shown partly)