Xi'an Jiaotong-Liverpool University
西交利物浦大學

XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	Database Development and Design (CPT201TC)
School Title	School of AI and Advanced Computing
Assignment Title	001: Assessment Task 1 (CW)
Submission Deadline	17:00, 10th Dec (Friday)
Final Word Count	NA
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.	**Yes**

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
  My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use	
Student ID	**1930080**

Stage of Marking	**Marker Code**	**Learning Outcomes Achieved (F/P/M/D) (please modify as appropriate)**			**Final Score**
		A	**B**	**C**	
1st Marker – red pen					
Moderation – green pen	**IM Initials**	The original mark has been accepted by the moderator (please circle as appropriate):			Y / N
		Data entry and score calculation have been checked by another tutor (please circle):			Y
2nd Marker if needed – green pen					

For Academic Office Use			**Possible Academic Infringement (please tick as appropriate)**	
Date Received	**Days late**	**Late Penalty**	☐ **Category A**	Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____
			☐ **Category B**	
			☐ **Category C**	
			☐ **Category D**	

			☐ **Category E**	

Students

The assignment must be submitted in a single pdf document via Learning Mall Online to the correct drop box. Only electronic submission is accepted and no hard copy submission.

All students must download their file and check that it is viewable after submission. Documents may become corrupted during the uploading process (e.g. due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for assessments.

- Code
- Screenshot
  - course_instructor table
  - course table
  - user table

Q4(c)

i.
- Code
- Result

ii.
- Code
- Result

iii.
- Code
- Result

iv.
- Code
- Result

v.
- Code
- Result

5. Data Warehousing and OLAP

i.
- Query Code
- Result

ii.
- Query Code
- Result

iii.
- Query Code
- Result

iv.
- Query Code
- Result

v.
- CUBE/GROUP BY

1. Advanced SQL, Triggers, and Indexing

Q1(a)

i.

Query statement

```
select distinct sname,major
from student inner join apply
on student.sid=apply.sid
order by sname asc
```

Result

SNAME	MAJOR
Amy	CS
Amy	EE
Bob	biology
Craig	CS
Craig	EE
Craig	bioengineering
Fay	history
Helen	CS
Irene	CS
Irene	biology
Irene	marine biology
Jay	history
Jay	psychology

ii.

Query statement

```
select sname,gpa,decision
from student inner join apply
on student.sid=apply.sid
where sizehs<1000
and apply.cname='Stanford'
and apply.major='CS'
```

Result

SNAME	GPA	DECISION
Helen	3.7	Y
Irene	3.9	N

iii.

Query statement

```
select abs(a.non_cs-b.cs_gpa) gpa
from (select avg(gpa) cs_gpa
from (select distinct student.sid,gpa
from student inner join apply
on student.sid=apply.sid
where student.sid=apply.sid
and apply.major ='CS')) b ,  (select avg(a.gpa) non_cs
from student a left join (select distinct student.sid,gpa
from student inner join apply
on student.sid=apply.sid
where student.sid=apply.sid
and apply.major ='CS') b
on a.sid=b.sid
where b.sid is null) a;
```

Result

GPA
.19428571428571428571428571428571428571

Download CSV

Q1(b)

Trigger

```
create or replace trigger student_after_insert
after insert on student
for each row
BEGIN
    if :new.gpa>3.3 and :new.gpa<=3.6 then
        INSERT INTO apply(SID,CNAME,MAJOR)
        VALUES(:new.sid,'Stanford','geology');
        INSERT INTO apply(SID,CNAME,MAJOR)
        VALUES(:new.sid,'MIT','biology');
    end if;
END;
```

Test SQL

```
select * from student;
select * from apply;
insert into Student values ('111', 'Kevin', 3.5, 1000);
insert into Student values ('222', 'Lori', 3.8, 1000);
select * from student;
select * from apply;
```

Result

SID	SNAME	GPA	SIZEHS
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
567	Edward	2.9	2000
678	Fay	3.8	200
789	Gary	3.4	800
987	Helen	3.7	800
876	Irene	3.9	400
765	Jay	2.9	1500
654	Amy	3.9	1000
543	Craig	3.4	2000

SID	CNAME	MAJOR	DECISION
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Cornell	history	N
765	Cornell	psychology	Y
543	MIT	CS	N

SID	SNAME	GPA	SIZEHS
111	Kevin	3.5	1000
222	Lori	3.8	1000
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
567	Edward	2.9	2000
678	Fay	3.8	200
789	Gary	3.4	800
987	Helen	3.7	800
876	Irene	3.9	400
765	Jay	2.9	1500
654	Amy	3.9	1000
543	Craig	3.4	2000

SID	CNAME	MAJOR	DECISION
111	Stanford	geology	-
111	MIT	biology	-
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Cornell	history	N
765	Cornell	psychology	Y
543	MIT	CS	N

Q1(c)

- `Student.sID, College.cName`

  This indexing method causes `College` to effectively scan `cornell` backward from the beginning, but Apply needs to be used in conjunction with nested loops. `College.cName` can be used as an index in order so that we are using the college name in the query then we can access the record quickly.

  Index for `college.cname` can provide a list of college which appear before `Cornell` in the list. `sID` will help us to list out records.

  `student.sID` and `college.cName` not only apply to where clause, but also can be applyed in college.cname< `cornell`

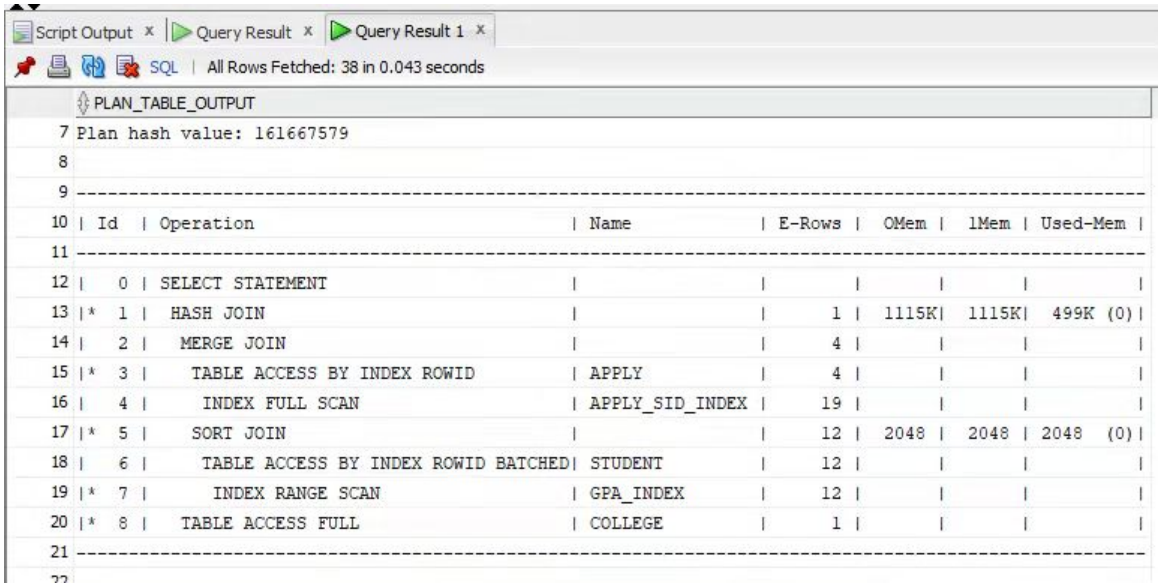
  And using creat index analysis block size of the first index:

Id	Operation	Name	E-Rows	OMem	1Mem	Used-Mem
0	SELECT STATEMENT					
1	NESTED LOOPS		1			
2	NESTED LOOPS		1			
*3	HASH JOIN		1	1506K	1506K	804K (0)
*4	TABLE ACCESS FULL	COLLEGE	1			
*5	TABLE ACCESS FULL	APPLY	4			
*6	INDEX RANGE SCAN	IDX_SID	1			
*7	TABLE ACCESS BY INDEX ROWID	STUDENT	1			

  The index will use fewer rows in searching and run time is 0.035 seconds

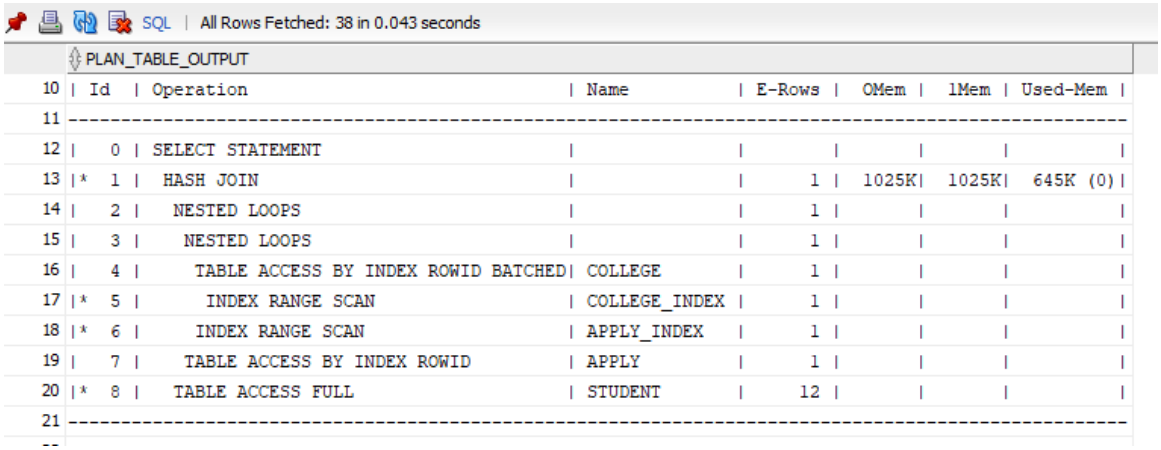
- `Student.sID, Student.GPA`

  If it is two separate indexes, `student.sID` will be useless. `Apply` and `College` require a large number of connections. and evaluate will GPA will be useful to retrieve rows with lesser number of data blocks access or not.



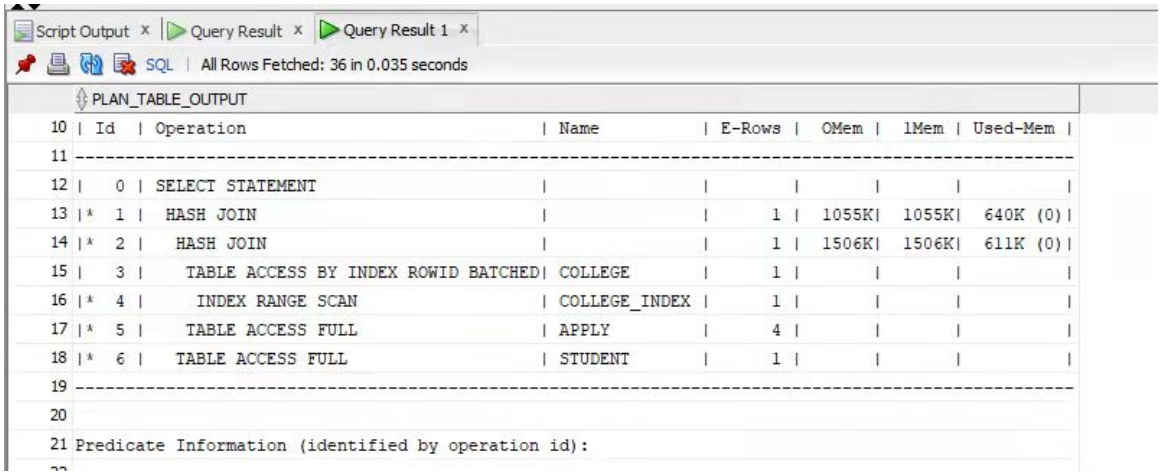
- `Apply.cName, College.cName`

  This may allow you to merge the two columns, but `Student` requires a big join.



- `Apply.sID,  Student.GPA`

  Student can be effectively scanned from 1.5, and Apply can also be searched, but College requires a large connection.



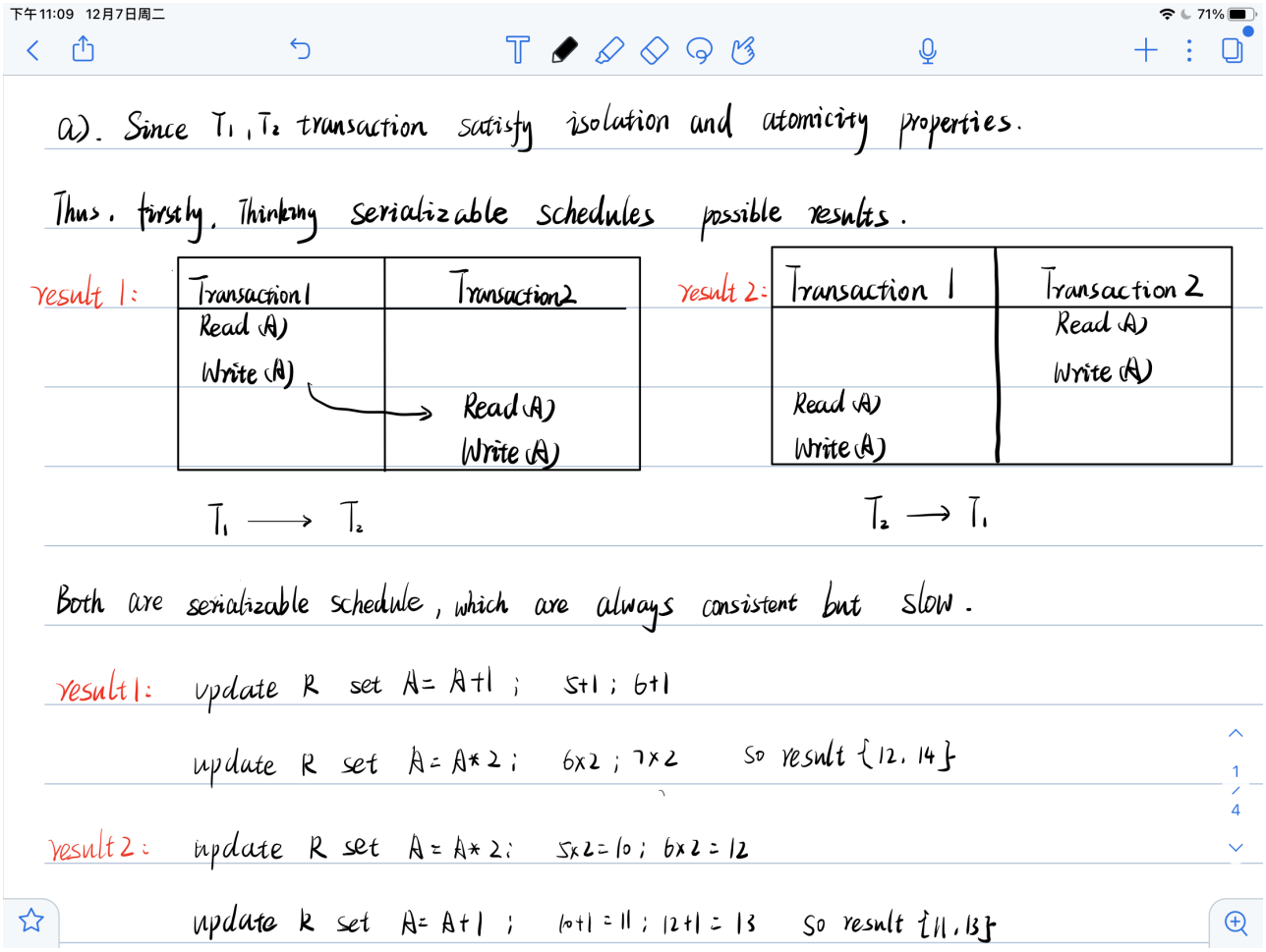
So choosing these two pairs：

- `Student.sID, College.cName`
- `Apply.sID,  Student.GPA`

of indexes will speed up the query time and you can see from the pictures that they use less disk space.

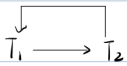
2. Transaction Management

Q2(a)



Secondly, non-serializable schedule

Transaction 1	Transaction 2
Read (A)	
	Read (A)
	Write (A)
Write (A)	

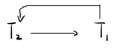


This shedule is not serializable as it has loop from T_2 to T_1 in the precedence graph.

result 3 : Update R set A = A+1

So result is {16}{17}.

Transaction 1	Transaction 2
	Read (A)
Read (A)	
Write (A)	
	Write (A)



result 4 : Update R set A = A*2

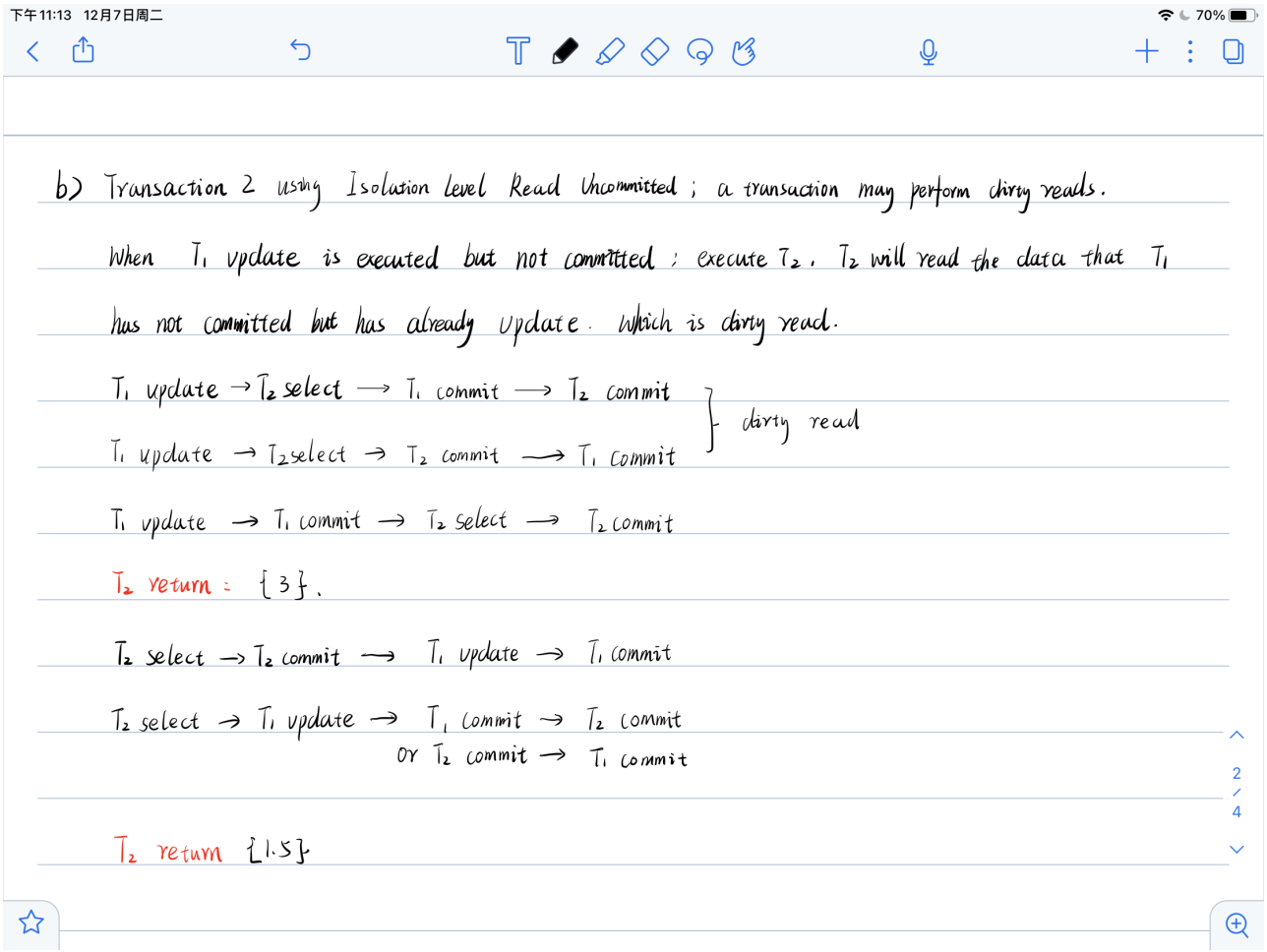
So result is {10}{12}.

Result 5: Since both transaction are submitted under the isolation and atomicity properties.

atomicity: Each transaction is "all-or-nothing" never left half done.

Transaction 1 Rollback : ⇒ T2 Rollback ; so result is {5, 6}.

Q2(b)



In addition, there are two possible scenarios for this question.

The table R(A) {(1),(2)}. only value 1 applyed T1 update. so the result is :

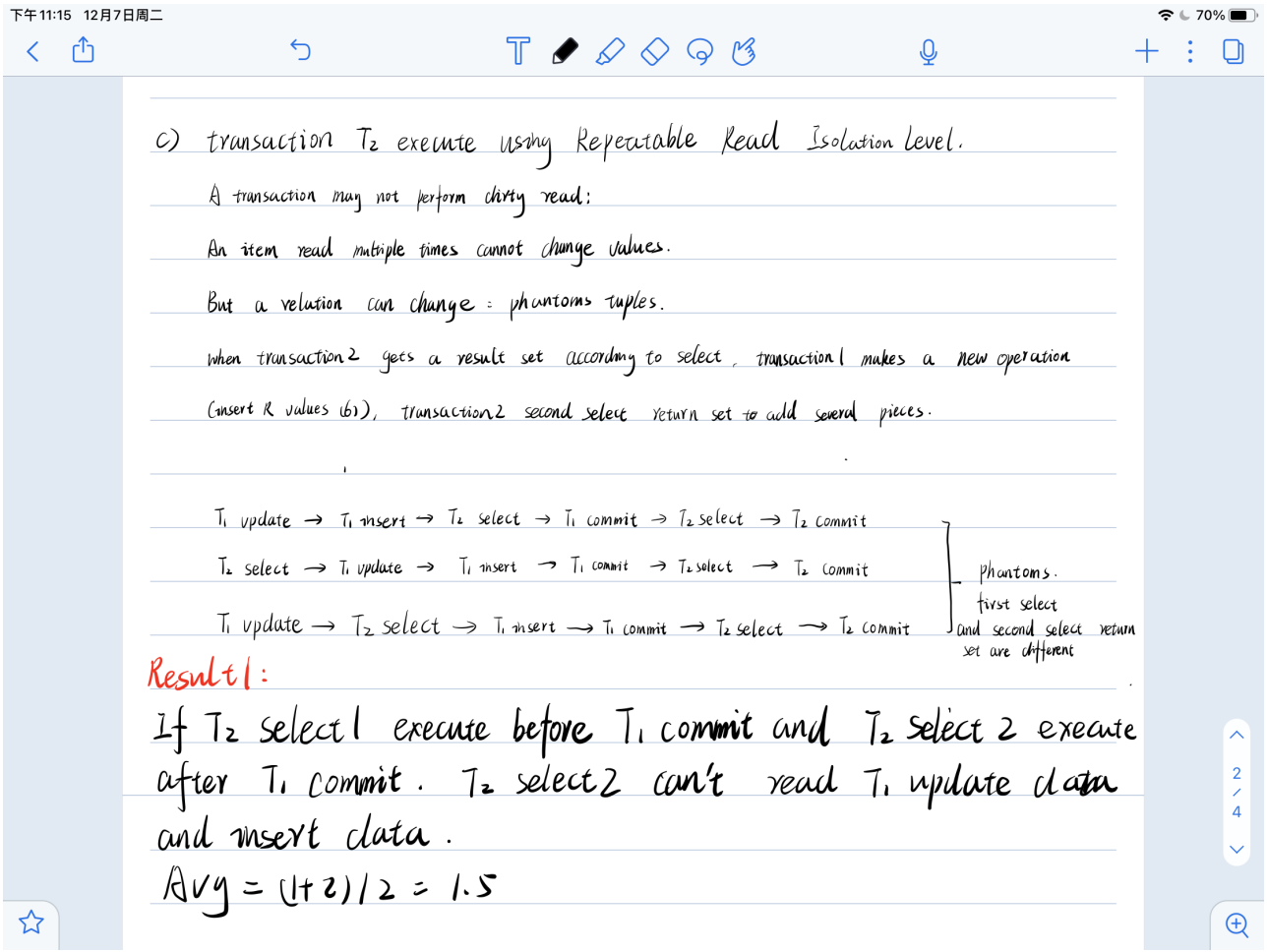
{2}

Or only value 2 applyed T1 update. so the result is:

{2.5}

Thus, results are {3},{1.5},{2},{2.5}

Q2(c)



c) transaction T_2 execute using Repeatable Read Isolation level.

A transaction may not perform dirty read;

An item read multiple times cannot change values.

But a relation can change : phantoms tuples.

When transaction 2 gets a result set according to select, transaction 1 makes a new operation (insert R values (6)), transaction 2 second select return set to add several pieces.

T_1 update → T_1 insert → T_2 select → T_1 commit → T_2 select → T_2 commit

T_2 select → T_1 update → T_1 insert → T_1 commit → T_2 select → T_2 commit

T_1 update → T_2 select → T_1 insert → T_1 commit → T_2 select → T_2 commit

} phantoms. first select and second select return set are different

Result1:

If T_2 select1 execute before T_1 commit and T_2 select 2 execute after T_1 commit. T_2 select2 can't read T_1 update data and insert data.

Avg = (1+2)/2 = 1.5

T_1 update → T_1 insert → T_1 commit → T_2 select → T_2 select → T_2 commit

Result2:

If T_1 execute and commit before T_2; T_2 select will read the data are updated.

Avg = (6(1+2)×2 + 6) / 3 = 4.

T_2 select — T_1 update — T_1 insert — T_2 select — T_1 commit — T_2 commit

Result3:

If T_1 update don't read by T_2, but T_1 insert do. so T_2 select2 will return:

Avg = (1+2+6) / 3 = 3

Q2(d)

The S1 is not recoverable, T2made a dirty read and committed before T1. When T1 execute rollback,T1 A gets its different to T2 value. Thus, DB in an inconsistent state. The final data B is different because the value of DB is submited based on T1. T2 and T3 has already utlized this wrong value and committed to the DB

The S2 is recoverable, S2's transaction3 need to be rolled back. None of the transaction has yet committed so the S2is recoverable. T2 does not include the B value, but T3 does, so T3rollback is required when T1 fails.

3. Querying XML Data

i.

Query Expression

```
<projects>
    {
    let $b := distinct-
values(/companyDB/employees/employee/projects/worksOn/@pno)
    for $p in $b
    order by number($p)
    return <project>
    {$p}
    </project>
    }
</projects>
```

Result

```
<?xml version="1.0" encoding="UTF-8"?>
<projects>
  <project>1</project>
  <project>2</project>
  <project>3</project>
  <project>10</project>
  <project>20</project>
  <project>30</project>
  <project>61</project>
  <project>62</project>
  <project>63</project>
  <project>91</project>
  <project>92</project>
</projects>
```

ii.

Query Expression

```
<department_manager>
{let $d:=doc("C:/XML/company.xml")
for $r in $d/companyDB/departments/department,
 $e in $d/companyDB/employees/employee
where $e/@ssn=$r/manager/@mssn
return
<department>{$r/dname}
<manager>{$e/lname}{$e/fname}</manager>
<salary>{$e/salary}</salary>
</department>
}
</department_manager>
```

Result

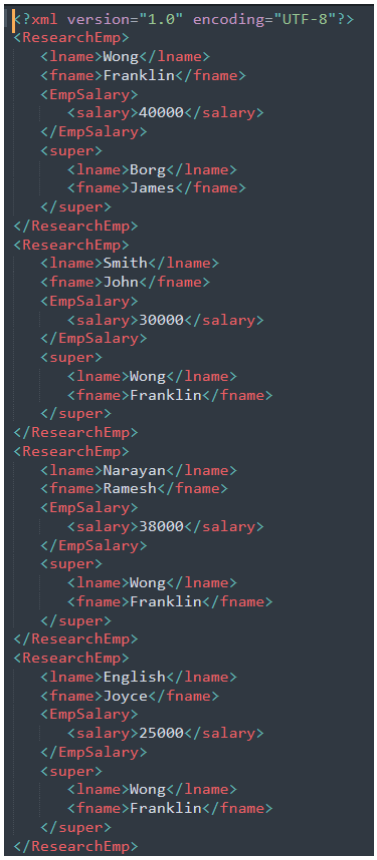
```
<?xml version="1.0" encoding="UTF-8"?>
<department_manager>
   <department>
      <dname>Headquarters</dname>
      <manager>
         <lname>Borg</lname>
         <fname>James</fname>
      </manager>
      <salary>
         <salary>55000</salary>
      </salary>
   </department>
   <department>
      <dname>Administration</dname>
      <manager>
         <lname>Wallace</lname>
         <fname>Jennifer</fname>
      </manager>
      <salary>
         <salary>43000</salary>
      </salary>
   </department>
   <department>
      <dname>Research</dname>
      <manager>
         <lname>Wong</lname>
         <fname>Franklin</fname>
      </manager>
      <salary>
         <salary>40000</salary>
      </salary>
   </department>
   <department>
      <dname>Software</dname>
      <manager>
         <lname>James</lname>
         <fname>Jared</fname>
      </manager>
      <salary>
         <salary>85000</salary>
      </salary>
   </department>
   <department>
      <dname>Hardware</dname>
      <manager>
         <lname>Freed</lname>
         <fname>Alex</fname>
      </manager>
      <salary>
         <salary>89000</salary>
      </salary>
   </department>
   <department>
      <dname>Sales</dname>
      <manager>
         <lname>James</lname>
         <fname>John</fname>
      </manager>
      <salary>
         <salary>81000</salary>
      </salary>
   </department>
</department_manager>
```

iii.

Query Expression

```
let $d:=doc("C:/XML/company.xml")
let $r:=$d/companyDB/departments/department[dname="Research"]
for $e in $d/companyDB/employees/employee,
$s in $d/companyDB/employees/employee
where $e/@worksFor=$r/@dno
and $e/@supervisor=$s/@ssn
return
<ResearchEmp>{$e/lname}{$e/fname}
<EmpSalary>{$e/salary}</EmpSalary>
<super>{$s/lname}{$s/fname}</super>
</ResearchEmp>
```

Result

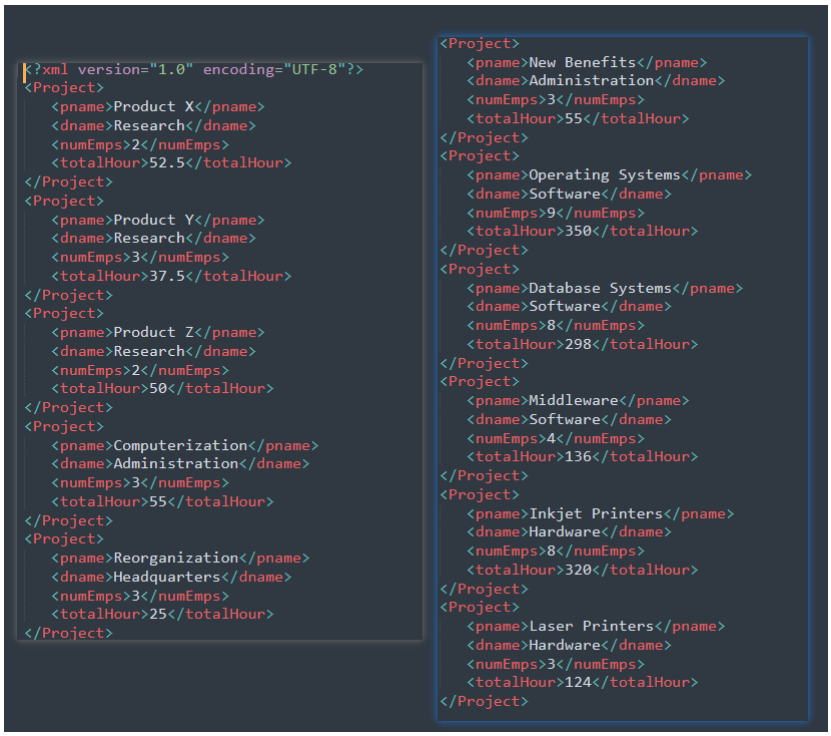


iv.

Query Expression

```
let $d:=doc("C:/XML/company.xml")
for $p in $d/companyDB/projects/project,
$x in $d/companyDB/departments/department
where $x/@dno = $p/@controllingDepartment
return
<Project>
{$p/pname}
{$x/dname}
<numEmps>
{count($p/workers/worker)}
</numEmps>
<totalHour>{sum($p/workers/worker)}</totalHour>
</Project>
```

Result



```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
   <pname>Product X</pname>
   <dname>Research</dname>
   <numEmps>2</numEmps>
   <totalHour>52.5</totalHour>
</Project>
<Project>
   <pname>Product Y</pname>
   <dname>Research</dname>
   <numEmps>3</numEmps>
   <totalHour>37.5</totalHour>
</Project>
<Project>
   <pname>Product Z</pname>
   <dname>Research</dname>
   <numEmps>2</numEmps>
   <totalHour>50</totalHour>
</Project>
<Project>
   <pname>Computerization</pname>
   <dname>Administration</dname>
   <numEmps>3</numEmps>
   <totalHour>55</totalHour>
</Project>
<Project>
   <pname>Reorganization</pname>
   <dname>Headquarters</dname>
   <numEmps>3</numEmps>
   <totalHour>25</totalHour>
</Project>
<Project>
   <pname>New Benefits</pname>
   <dname>Administration</dname>
   <numEmps>3</numEmps>
   <totalHour>55</totalHour>
</Project>
<Project>
   <pname>Operating Systems</pname>
   <dname>Software</dname>
   <numEmps>9</numEmps>
   <totalHour>350</totalHour>
</Project>
<Project>
   <pname>Database Systems</pname>
   <dname>Software</dname>
   <numEmps>8</numEmps>
   <totalHour>298</totalHour>
</Project>
<Project>
   <pname>Middleware</pname>
   <dname>Software</dname>
   <numEmps>4</numEmps>
   <totalHour>136</totalHour>
</Project>
<Project>
   <pname>Inkjet Printers</pname>
   <dname>Hardware</dname>
   <numEmps>8</numEmps>
   <totalHour>320</totalHour>
</Project>
<Project>
   <pname>Laser Printers</pname>
   <dname>Hardware</dname>
   <numEmps>3</numEmps>
   <totalHour>124</totalHour>
</Project>
```

V.

Query Expression

```
let $d:=doc("C:/XML/company.xml")
for $p in $d/companyDB/projects/project,
$x in $d/companyDB/departments/department
where $x/@dno = $p/@controllingDepartment
and count($p/workers/worker)>1
return
<Project>
{$p/pname}
{$x/dname}
<numEmps>
{count($p/workers/worker)}
</numEmps>
<totalHour>{sum($p/workers/worker)}</totalHour>
</Project>
```

Result

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
   <pname>Product X</pname>
   <dname>Research</dname>
   <numEmps>2</numEmps>
   <totalHour>52.5</totalHour>
</Project>
<Project>
   <pname>Product Y</pname>
   <dname>Research</dname>
   <numEmps>3</numEmps>
   <totalHour>37.5</totalHour>
</Project>
<Project>
   <pname>Product Z</pname>
   <dname>Research</dname>
   <numEmps>2</numEmps>
   <totalHour>50</totalHour>
</Project>
<Project>
   <pname>Computerization</pname>
   <dname>Administration</dname>
   <numEmps>3</numEmps>
   <totalHour>55</totalHour>
</Project>
<Project>
   <pname>Reorganization</pname>
   <dname>Headquarters</dname>
   <numEmps>3</numEmps>
   <totalHour>25</totalHour>
</Project>
<Project>
   <pname>New Benefits</pname>
   <dname>Administration</dname>
   <numEmps>3</numEmps>
   <totalHour>55</totalHour>
</Project>
<Project>
   <pname>Operating Systems</pname>
   <dname>Software</dname>
   <numEmps>9</numEmps>
   <totalHour>350</totalHour>
</Project>
<Project>
   <pname>Database Systems</pname>
   <dname>Software</dname>
   <numEmps>8</numEmps>
   <totalHour>298</totalHour>
</Project>
<Project>
   <pname>Middleware</pname>
   <dname>Software</dname>
   <numEmps>4</numEmps>
   <totalHour>136</totalHour>
</Project>
<Project>
   <pname>Inkjet Printers</pname>
   <dname>Hardware</dname>
   <numEmps>8</numEmps>
   <totalHour>320</totalHour>
</Project>
<Project>
   <pname>Laser Printers</pname>
   <dname>Hardware</dname>
   <numEmps>3</numEmps>
   <totalHour>124</totalHour>
</Project>
```

4. Object-Relational Database

Q4(a)

```
class Course(models.Model):
    name = models.CharField(null=False,max_length=100,default='online course')
    description = models.CharField(max_length=500)
    instructors = models.ManyToManyField(Instructor)
    learners = models.ManyToManyField(Learner,through='enrollment')

    def __str__(self):
        return "Name: "+self.name+", "+\
            "Description: "+self.description
```

Q4(b)

Code

```
def write_course_instructor_relationships():
    # Get related instructors
    instructor_yan = Instructor.objects.get(first_name='Yan')
    instructor_joy = Instructor.objects.get(first_name='Joy')
    instructor_peter = Instructor.objects.get(first_name='Peter')

    # Get related courses
    course_cloud_app = Course.objects.get(name__contains='Cloud')
    course_python = Course.objects.get(name__contains='Python')

    # Add instructors to courses
    course_cloud_app.instructors.add(instructor_yan)
    course_cloud_app.instructors.add(instructor_joy)
    course_python.instructors.add(instructor_peter)

    print("Course-instructor relationships saved... ")
```

Screenshot

course_instructor table

表(T)： CRUD_course_instructors

	id	course_id	instructor_id
	…	过滤	过滤
1	1	1	3
2	2	1	4
3	3	2	5

course table

(T) : CRUD_course 在所有列中过滤

id	name	description
···	过滤	过滤
1	Cloud Application Development with Database	Develop and deploy application on cloud
2	Introduction to Python	Learn core concepts of Python and obtain hands-on ···

user table

	id	first_name	last_name	dob
	···	过滤	过滤	过滤
1	1	John	Doe	1962-07-16
2	2	John	Doe	*NULL*
3	3	Yan	Luo	1962-07-16
4	4	Joy	Li	1992-01-02
5	5	Peter	Chen	1982-05-02
6	6	James	Smith	1982-07-16
7	7	Mary	Smith	1991-06-12
8	8	Robert	Lee	1999-01-02
9	9	David	Smith	1983-07-16
10	10	John	Smith	1986-03-16

Q4(c)

i.

Code

```
 # Find students with last name "Smith"
learners_smith = Learner.objects.filter(last_name='Smith')
print("1. Find learners with last name `Smith`")
for learner in learners_smith:
    print(learner)
print("\n")
```

Result

```
1. Find learners with last name `Smith`
First name: James, Last name: Smith, Date of Birth: 1982-07-16, Occupation: data_scientist, Social Link: https://www.linkedin.com/james/
First name: Mary, Last name: Smith, Date of Birth: 1991-06-12, Occupation: dba, Social Link: https://www.facebook.com/mary/
First name: David, Last name: Smith, Date of Birth: 1983-07-16, Occupation: developer, Social Link: https://www.linkedin.com/david/
First name: John, Last name: Smith, Date of Birth: 1986-03-16, Occupation: developer, Social Link: https://www.linkedin.com/john/
```

ii.

Code

```
# Order by dob descending, and select the first two objects
learners = Learner.objects.order_by("-dob" )[0:2]
print("2. Find top two youngest learners")
for learner in learners:
    print(learner)
print("\n")
```

Result

```
2. Find top two youngest learners
First name: Robert, Last name: Lee, Date of Birth: 1999-01-02, Occupation: student, Social Link: https://www.facebook.com/robert/
First name: Mary, Last name: Smith, Date of Birth: 1991-06-12, Occupation: dba, Social Link: https://www.facebook.com/mary/
```

iii.

Code

```
course=Course.objects.get(name='Cloud Application Development with Database')
learners_db=course.learners.all()
print("iii. Retrieve all learners for the Cloud Application Development with Database course.")
for learner in learners_db:
    print(learner)
print("\n")
```

Result

```
iii.    Retrieve all learners for the Cloud Application Development with Database course.
First name: James, Last name: Smith, Date of Birth: 1982-07-16, Occupation: data_scientist, Social Link: https://www.linkedin.com/james/
First name: Mary, Last name: Smith, Date of Birth: 1991-06-12, Occupation: dba, Social Link: https://www.facebook.com/mary/
First name: David, Last name: Smith, Date of Birth: 1983-07-16, Occupation: developer, Social Link: https://www.linkedin.com/david/
First name: John, Last name: Smith, Date of Birth: 1986-03-16, Occupation: developer, Social Link: https://www.linkedin.com/john/
```

iv.

Code

```
instructors=Instructor.objects.filter(course__name="Introduction to Python")
print("iv.  Retrieve instructors for “introduction to python” course.")
for instructor in instructors:
    print(instructor)
print("\n")
```

Result

```
iv.     Retrieve instructors for “introduction to python” course.
First name: Peter, Last name: Chen, Is full time: True, Total Learners: 2002
```

v.

Code

```
courses=Course.objects.filter(instructors__first_name='Peter')
occupation_list=set()
for course in courses:
    for learner in course.learners.all():
        occupation_list.add(learner.occupation)
print("v.Retrieve occupation list of learners for the courses taught by instructor “Peter”.")
for occupation in occupation_list:
    print(occupation)
```

Result

```
v.Retrieve occupation list of learners for the courses taught by instructor “Peter”.
student
-
```

5. Data Warehousing and OLAP

i.

Query Code

```
SELECT StoreZip, TimeMonth, SUM(SalesDollar) AS SumSales,
       MIN(SalesDollar) AS MinSales, COUNT(salesdollar) AS
RowCount
 FROM SSSales, SSStore, SSTimeDim
 WHERE SSSales.StoreId = SSStore.StoreId
   AND SSSales.TimeNo = SSTimeDim.TimeNo
   AND (StoreNation = 'USA' OR StoreNation = 'Canada')
   AND TimeYear = 2016
 GROUP BY (StoreZip, TimeMonth)
 ORDER BY StoreZip, TimeMonth;
```

Result

STOREZIP	TIMEMONTH	SUMSALES	MINSALES	ROWCOUNT
80111-0033	2	10390	5195	2
80111-0033	5	12420	6210	2
80111-0033	7	9630	4815	2
80111-0033	10	10616	5308	2
80129-5543	2	21460	5215	4
80129-5543	5	24460	6015	4
80129-5543	7	23660	5915	4
80129-5543	10	23180	5745	4
98104-2211	2	20640	5115	4
98104-2211	5	24840	6115	4
98104-2211	7	23596	5844	4
98104-2211	10	26040	6245	4

ii.

Query Code

```
SELECT StoreZip, TimeMonth, SUM(SalesDollar) AS SumSales,
       MIN(SalesDollar) AS MinSales, COUNT(salesdollar) AS
RowCount
 FROM SSSales, SSStore, SSTimeDim
 WHERE SSSales.StoreId = SSStore.StoreId
   AND SSSales.TimeNo = SSTimeDim.TimeNo
   AND (StoreNation = 'USA' OR StoreNation = 'Canada')
   AND TimeYear = 2016
 GROUP BY CUBE(StoreZip, TimeMonth)
 ORDER BY StoreZip, TimeMonth;
```

Result

STOREZIP	TIMEMONTH	SUMSALES	MINSALES	ROWCOUNT
80111-0033	2	10390	5195	2
80111-0033	5	12420	6210	2
80111-0033	7	9630	4815	2
80111-0033	10	10616	5308	2
80111-0033	-	43056	4815	8
80129-5543	2	21460	5215	4
80129-5543	5	24460	6015	4
80129-5543	7	23660	5915	4
80129-5543	10	23180	5745	4
80129-5543	-	92760	5215	16
98104-2211	2	20640	5115	4
98104-2211	5	24840	6115	4
98104-2211	7	23596	5844	4
98104-2211	10	26040	6245	4
98104-2211	-	95116	5115	16
-	2	52490	5115	10
-	5	61720	6015	10
-	7	56886	4815	10
-	10	59836	5308	10
-	-	230932	4815	40

iii.

Query Code

```
SELECT TimeYear, TimeMonth, SUM(SalesDollar) AS SumSales,
       MIN(SalesDollar) AS MinSales, COUNT(salesdollar) AS
RowCount
 FROM SSSales, SSStore, SSTimeDim
 WHERE SSSales.StoreId = SSStore.StoreId
   AND SSSales.TimeNo = SSTimeDim.TimeNo
   AND (StoreNation = 'USA' OR StoreNation = 'Canada')
   AND TimeYear between 2016 and 2017
 GROUP BY rollup(timeyear, TimeMonth)
 ORDER BY timeyear, TimeMonth;
```

Result

TIMEYEAR	TIMEMONTH	SUMSALES	MINSALES	ROWCOUNT
2016	2	52490	5115	10
2016	5	61720	6015	10
2016	7	56886	4815	10
2016	10	59836	5308	10
2016	-	230932	4815	40
2017	2	74910	5055	14
2017	5	89110	6005	14
2017	7	75086	4605	14
2017	10	81288	5448	14
2017	-	320394	4605	56
-	-	551326	4605	96

iv.

Query Code

```
SELECT TimeYear,Timequarter, TimeMonth, SUM(SalesDollar) AS SumSales,
       MIN(SalesDollar) AS MinSales, COUNT(salesdollar) AS RowCount
 FROM SSSales, SSStore, SSTimeDim
 WHERE SSSales.StoreId = SSStore.StoreId
   AND SSSales.TimeNo = SSTimeDim.TimeNo
   AND (StoreNation = 'USA' OR StoreNation = 'Canada')
   AND TimeYear between 2016 and 2017
 GROUP BY rollup(timeyear,timequarter,TimeMonth)
 ORDER BY timeyear,timequarter, TimeMonth;
```

Result

TIMEYEAR	TIMEQUARTER	TIMEMONTH	SUMSALES	MINSALES	ROWCOUNT
2016	1	2	26245	5115	5
2016	1	-	26245	5115	5
2016	2	5	30860	6015	5
2016	2	-	30860	6015	5
2016	3	7	28443	4815	5
2016	3	-	28443	4815	5
2016	4	10	29918	5308	5
2016	4	-	29918	5308	5
2016	-	-	115466	4815	20
2017	1	2	37455	5055	7
2017	1	-	37455	5055	7
2017	2	5	44555	6005	7
2017	2	-	44555	6005	7
2017	3	7	37543	4605	7
2017	3	-	37543	4605	7
2017	4	10	40644	5448	7
2017	4	-	40644	5448	7
2017	-	-	160197	4605	28
-	-	-	275663	4605	48

v.

CUBE/GROUP BY

STOREZIP	TIMEMONTH	SUMSALES	MINSALES	ROWCOUNT
80111-0033	2	10390	5195	2
80111-0033	5	12420	6210	2
80111-0033	7	9630	4815	2
80111-0033	10	10616	5308	2
80129-5543	2	21460	5215	4
80129-5543	5	24460	6015	4
80129-5543	7	23660	5915	4
80129-5543	10	23180	5745	4
98104-2211	2	20640	5115	4
98104-2211	5	24840	6115	4
98104-2211	7	23596	5844	4
98104-2211	10	26040	6245	4

STOREZIP	TIMEMONTH	SUMSALES	MINSALES	ROWCOUNT
80111-0033	2	10390	5195	2
80111-0033	5	12420	6210	2
80111-0033	7	9630	4815	2
80111-0033	10	10616	5308	2
80111-0033	-	43056	4815	8
80129-5543	2	21460	5215	4
80129-5543	5	24460	6015	4
80129-5543	7	23660	5915	4
80129-5543	10	23180	5745	4
80129-5543	-	92760	5215	16
98104-2211	2	20640	5115	4
98104-2211	5	24840	6115	4
98104-2211	7	23596	5844	4
98104-2211	10	26040	6245	4
98104-2211	-	95116	5115	16
-	2	52490	5115	10
-	5	61720	6015	10
-	7	56886	4815	10
-	10	59836	5308	10
-	-	230932	4815	40

The `cube` operator produce all possible subtotal combinations in addiition to the normal totals shown in a group by clause

Groupby simply groups storezip and timemonth. This may not meet the demand in the business.
But cube carries out total for each type of data, sum for each type of storezip, and sum for each type of timemonth. Also sum all the data.

This is of great benefit to the business, for example, you can view the results of a given quarter or the sum of business requirements. A number of new tuples will be generated.

TIMEYEAR	TIMEMONTH	SUMSALES	MINSALES	ROWCOUNT
2016	2	52490	5115	10
2016	5	61720	6015	10
2016	7	56886	4815	10
2016	10	59836	5308	10
2016	-	230932	4815	40
2017	2	74910	5055	14
2017	5	89110	6005	14
2017	7	75086	4605	14
2017	10	81288	5448	14
2017	-	320394	4605	56
-	-	551326	4605	96

`ROLLUP` operator is partial set of subtotals, and appropriate for hierarchical dimensions.

`ROLLUP` only subtotal for `timeyear` instead of all possible subtotals for `(timeyear,timemonth)`. Howerever, this type operator can view some specific results of sum for business requirements.