CSC2001F Assignment 4 Report

Alex Soldin

SLDALE003

Section A

1

For each of the 3 relations state what attribute(s) you would use as its primary key

Relation Name	Primary Key
codeCat	code
spend	id, code, transactionDate
account	id

To get a unique primary key for the spend relation, I considered id, code and transactionDate as grouping these three attributes together guarantees completely unique tuples.

2

List all the foreign keys in your 3 relations

spend.id = account.id
spend.code = codeCat.code

Section B

1

Show any one complete tuple (just 1, no more) from each of your 3 relations.

```
select * from codeCat limit 1;
+-----------------+------+
| category        | code |
+-----------------+------+
| Gifts_Donations | 8661 |
+-----------------+------+
```

```
select * from spend limit 1;
+-------------+-----------------+---------+-----------------------+-------+
| id          | transactionDate | amount  | merchant              | code  |
+-------------+-----------------+---------+-----------------------+-------+
| 12291840250 | 2018/02/01 1:23 | -606.15 | 868652 BLOEMFONTEIN 1 | 5541  |
+-------------+-----------------+---------+-----------------------+-------+
```

```
select * from account limit 1;
+-------------+-----------+---------+-----+---------------+--------+
| id          | firstName | surname | age | maritalStatus | income |
+-------------+-----------+---------+-----+---------------+--------+
|   291311668 | Fani      | Raziya  |  98 | D             | low    |
+-------------+-----------+---------+-----+---------------+--------+
```

2

Output the number of tuples in each of your 3 relations.

```
select count(category) from codeCat;
```

```
+-----------------+
| count(category) |
+-----------------+
|             254 |
+-----------------+
```

```
select count(id) from spend;
+-----------+
| count(id) |
+-----------+
|      5906 |
+-----------+
```

```
select count(id) from account;
+-----------+
| count(id) |
+-----------+
|      1247 |
+-----------+
```

3

List first name and surname of everyone with surname “Naidoo”, in alphabetical order of first name.

```
select firstName, surname from account where surname = 'Naidoo' order by
firstName asc;
```

```
+-------------+---------+
| firstName   | surname |
+-------------+---------+
| Kemuel      | Naidoo  |
| Kenneth     | Naidoo  |
| Kevendren   | Naidoo  |
| Kgosietsile | Naidoo  |
| Kgothalang  | Naidoo  |
| Kgotsofatso | Naidoo  |
| Khanyisa    | Naidoo  |
| Khanyisani  | Naidoo  |
| Kharan      | Naidoo  |
| Kholofelo   | Naidoo  |
| Khomotso    | Naidoo  |
+-------------+---------+
```

4

For each code 5511 transaction, give the transactionDate along with the VAT on that transaction, where the VAT was 15% of the amount spent (e.g. if the amount was -115 then the VAT was 15).

```
select transactionDate, (-1)*amount*(1 - 1/1.15) as VAT from spend where
code = 5511;
```

```
+------------------+--------------------+
| transactionDate  | VAT                |
+------------------+--------------------+
| 2018/02/01 14:05 |      15.000000045 |
| 2018/02/02 13:31 |       46.95652188 |
| 2018/02/04 10:43 |    59.99347844085 |
| 2018/02/05 6:45  | 71.73913064999999 |
| 2018/02/06 8:14  |      10.956521772 |
| 2018/02/06 11:57 |       326.0869575 |
+------------------+--------------------+
```

5

Show the firstname, surname, age, marital-status and income category of everyone who has any missing data in any of these fields.

```
select firstName, surname, age, maritalStatus, income from account where
firstName is null or surname is null or age is null or maritalStatus is
null or income is null;
```

```
Empty set
```

This result is expected as when the tables were created, it was specified that each field was only to be included if it was not null. For example, the firstName field was created using the following statement:
Create table if not exists account (firstName VARCHAR(20) NOT NULL);

6

Give the names of merchants who have had transactions of both codes 5111 and 2741.

```
select distinct merchant from Spend where code = 5111 and merchant = some
(select merchant from Spend where code = 2741);
```

```
+------------------------+
| merchant               |
+------------------------+
| 3 @ 1                  |
+------------------------+
```

7

Find the transactionDate and merchant associated with the highest amount in the database, along with that amount.

```
select transactionDate, merchant, -1*amount as HighestAmount from spend
where amount = (select min(amount) from spend);
```

OR

```
select transactionDate, merchant, -1*amount as HighestAmount from spend
order by -1*amount desc limit 1;
```

```
+----------------+--------------------+---------------+
| transactionDate | merchant           | HighestAmount |
+----------------+--------------------+---------------+
| 2018/02/03 7:13 | @Home Secunda Mall |      18385.09 |
+----------------+--------------------+---------------+
```

8

How many people (ids) have any transaction amount that is higher than every amount ever spent by the person with id 12312870316 ?

```
select count(id) from spend where -1*amount > (select -1*amount from spend
where id = '12312870316');
```

```
+-----------+
| count(id) |
+-----------+
|         9 |
+-----------+
```

9

How many categories are there? (i.e. how many different category values?)

```
select count(distinct category) as NumberOfCategories from codeCat;
```

```
+--------------------+
| NumberOfCategories |
+--------------------+
|                 12 |
+--------------------+
```

10

Give the average transaction amount (a single value in the answer).

```
select avg(-1*amount) as Average from spend;
```

```
+--------------------+
| Average            |
+--------------------+
| 275.44979004402603 |
+--------------------+
```

11

Find the code range for each category (i.e. each result line will have category name, lowest code and highest code for that category).

```
select category, max(code) as HighestCode, min(code) as LowestCode from
codeCat group by category;
```

```
+-----------------+-------------+------------+
| category        | HighestCode | LowestCode |
+-----------------+-------------+------------+
| Clothing        |        7296 |       5137 |
| Eat_Out         |        7311 |       5462 |
| Education       |        9402 |       2741 |
| Entertainment   |        7995 |       4899 |
| Gifts_Donations |        8661 |       5045 |
| Groceries       |        5451 |       2000 |
| Health_Fitness  |        8699 |       5310 |
| Holiday_Travel  |        7991 |       3000 |
| Home            |        9399 |        763 |
| Medical         |        8099 |       5047 |
| Pets            |        7998 |        742 |
| Transport       |        7549 |       4784 |
+-----------------+-------------+------------+
```

12

Find the largest transaction amount for each code that is associated with more than 50 transactions.

```
select code, max(-1*amount) as HighestAmount from spend group by code
having count(transactionDate) > 50;
```

```
+------+---------------+
| code | HighestAmount |
+------+---------------+
| 5111 |         617.8 |
| 5499 |          3410 |
| 5541 |        2000.1 |
| 5712 |          6000 |
| 5719 |      18385.09 |
| 5812 |          3690 |
| 5912 |          7000 |
| 5943 |        1910.5 |
| 7011 |        2876.4 |
+------+---------------+
```

13

Which code(s) have the least number of transactions i.e. which code(s) have the fewest transactions?

```
select code, count(transactionDate) as NumberOfTransactions from spend
group by code having NumberOfTransactions = (select count(transactionDate)
as transactions from spend group by code order by transactions asc limit
1);
```

```
+------+----------------------+
| code | NumberOfTransactions |
+------+----------------------+
| 3405 |                    1 |
| 4900 |                    1 |
| 5013 |                    1 |
| 5039 |                    1 |
| 5047 |                    1 |
| 5309 |                    1 |
| 5621 |                    1 |
| 5661 |                    1 |
| 5947 |                    1 |
| 5971 |                    1 |
| 7641 |                    1 |
| 7692 |                    1 |
| 7991 |                    1 |
| 8043 |                    1 |
+------+----------------------+
```

14

Show all information on all large transactions including the category involved. A large transaction is one with an amount that exceeds ten times the average transaction amount (e.g. if the average transaction amount is R100, a large transaction has an amount above R1000).

```
select spend.*, category from codeCat, spend where spend.code =
codeCat.code and (spend.amount) < 10*(select avg(amount) from spend);
```

id	transactionDate	amount	merchant	code	category
12297034299	2018/02/01 10:09	-5000	200 Baviaanspoortweg{#	7911	Entertainment
12297272968	2018/02/01 10:33	-3545	1 STOP MOTOR SPARES	5533	Transport
12299530095	2018/02/01 14:04	-2876.4	54 ON BATH	7011	Holiday_Travel
12291743683	2018/02/01 20:10	-3690	5TH AVE..	5812	Gifts_Donations
12312298975	2018/02/02 7:43	-2770	4 YOU HARDWARE NQAMAKW	5251	Home
12312870316	2018/02/02 8:41	-5000	4 YOU HARDWARE NQAMAKW	5251	Home
12313442532	2018/02/02 9:30	-3999.9	849 - BT GAMES CLEARWA	7993	Entertainment
12314550122	2018/02/02 11:08	-5299.9	849 - BT GAMES ILANGA	7993	Entertainment
12314566711	2018/02/02 11:09	-3410	382522 ULTRA CITY PIET	5499	Gifts_Donations
12315064423	2018/02/02 11:52	-2999.7	849 - BT GAMES MIMOSA	7993	Entertainment
12326817964	2018/02/03 7:13	-18385.09	@Home Secunda Mall	5719	Home
12327008160	2018/02/03 7:32	-8152.44	000702 PENNY PINCHERS	5211	Home
12329107529	2018/02/03 9:32	-4310.56	24 LADYMEAD	3405	Holiday_Travel
12328755357	2018/02/03 10:04	-6000	4 D BEDDING CC	5712	Eat_Out
12330362338	2018/02/03 12:22	-5900	4 YOU HARDWARE WILLOWV	5251	Home
12349286107	2018/02/05 9:56	-3000	5 STAR FURNITURE PTA	5021	Home
12349843967	2018/02/05 11:02	-4511	3 START HARDWARE OLIVE	5251	Home
12359393961	2018/02/06 11:11	-6299.8	849 - BT GAMES MIMOSA	7993	Entertainment
12359703132	2018/02/06 11:55	-8000	*KK ROOFSHEETING CC	1761	Home
12376294619	2018/02/08 6:07	-7000	*NETCARE HOLDINGS (PTY	5912	Health_Fitness
12379967451	2018/02/08 15:38	-5999	@Home Centurion 2	5719	Home

15

Find the total amount spent on Pets for each merchant where there have been more than 10 transactions in the Pets category.

```
select merchant, sum(-1*amount) as Total from spend, codeCat where
codeCat.code = spend.code and codeCat.category = 'Pets' group by merchant
having count(transactionDate) > 10;
```

merchant	Total
*WALKERVILLE FEEDS CC	5026
3005 CHECK STAR DURBAN	3955.89
3R DIEREVET	7394.4
9th Avenue Vet Clinic	5468.740000000001

16

For each category, show how many such transactions there are in the database along with the total Rand amount of those transactions.

```
select category, count(transactionDate) as NumberOfTransactions, sum(-
1*amount) as TotalRandAmount from spend, codeCat where codeCat.code =
spend.code group by category;
```

category	NumberOfTransactions	TotalRandAmount
Clothing	12	7000
Eat_Out	122	67420.8
Education	237	36653.09
Entertainment	51	43350.61000000003
Gifts_Donations	599	49849.72000000012
Groceries	3	765.0999999999999
Health_Fitness	4340	1123170.8700000073
Holiday_Travel	135	72976.86
Home	237	167686.18999999997
Medical	31	8101.91
Pets	101	31514.480000000007
Transport	38	18316.83

Section C

Determine the amount for each income group and the total number of account holders, total number of single people, total number of marries people, total number of divorced people and the total number of widowed people.

The statement above needs to display the average amount spent by each income group as well as displays the number of people that are single, married, divorce and windowed people within this income group.

I know this result is correct because:

- The total number of people per income category is equal to the sum of the number of single, married, divorced and widowed account holders.
- The 3 averages can be added together and divided by 3 yielding the same result as averaging all the transaction amounts (as seen in question 10 above).
- There are only 3 income classes defined and therefore only 3 tuples.

```
select income as Income, avg(-1*amount) as Average, count(account.id) as
TotalPerIncome, count(case when maritalStatus = 'S' then 1 else null end)
as Single, count(case when maritalStatus = 'M' then 1 else null end) as
Married, count(case when maritalStatus = 'D' then 1 else null end) as
Divorced, count(case when maritalStatus = 'W' then 1 else null end) as
Widowed from account, spend where account.id = spend.id group by income
order by Average;
```

```
+--------+--------------------+---------------+--------+---------+----------+---------+
| Income | Average            | TotalPerIncome | Single | Married | Divorced | Widowed |
+--------+--------------------+---------------+--------+---------+----------+---------+
| low    | 233.79245412844028 |           436 |    151 |     143 |      115 |      27 |
| avg    |  289.1447009966776 |           602 |    126 |     235 |      174 |      67 |
| high   |  298.5087793427231 |           213 |     43 |      93 |       51 |      26 |
+--------+--------------------+---------------+--------+---------+----------+---------+
```

Section D

1

Delete all transactions from merchant 3@1 (however it may be spelt).

```
delete from spend where merchant like "3%@%1%" or merchant like "3%AT%1%";
```

```
Query OK, 167 rows affected
```

Check

```
select * from spend where merchant like "3%@%1%" or merchant like
"3%AT%1%";
Empty set
```

2

Insert a new transaction for id 12393560590 that took place at midnight on New Year's Eve 2018 for an amount of R448. The code is 5251 but the merchant and description are not known.

```
insert into spend (id, transactionDate, amount, merchant, code) values
("12393560590", "2019/01/01 0:00" , -448, "", "5251");
```

```
Query OK, 1 row affected
```

Check

```
select * from spend where id = "12393560590";
+-------------+-----------------+--------+----------+------+
| id          | transactionDate | amount | merchant | code |
+-------------+-----------------+--------+----------+------+
| 12393560590 | 2019/01/01 0:00 |   -448 |          | 5251 |
+-------------+-----------------+--------+----------+------+
```

3

Code 5211 purchases should have been coded 5221, so change 5211 to 5221.

```
update spend set code = 5221 where code = 5211;
```

```
Query OK, 25 rows affected
Rows matched: 25  Changed: 25  Warnings: 0
```

Check

```
select * from spend where code = "5211";
Empty set
```

```
select * from spend where code = "5221";
+-------------+------------------+----------+-----------------------+------+
| id          | transactionDate  | amount   | merchant              | code |
+-------------+------------------+----------+-----------------------+------+
| 12295108452 | 2018/02/01 6:39  |  -429.99 | 747636 PENNYPINCHERS  | 5221 |
| 12297316953 | 2018/02/01 10:38 |  -137.46 | 747636 PENNYPINCHERS  | 5221 |
| 12299545090 | 2018/02/01 14:05 |  -127.13 | 000660 PENNYPINCHERS R | 5221 |
```

(Only showing the first 3 rows)

4

For everyone who is not M(married), add another row for them, which has their maritalStatus as U (unmarried) and has the values in all the other columns of their new extra row the same as in their original row.

```
insert into account (id, firstName, surname, age, maritalStatus, income)
select id, firstName, surname, age, "U", income from account where
maritalStatus <> "M";
```

```
Query OK, 777 rows affected
Records: 777  Duplicates: 0  Warnings: 0
```

Check

```
select count(*) as Total, count(case when maritalStatus = 'M' then 1 else
null end) as Married, count(case when maritalStatus <> 'U' and
maritalStatus <> 'M' then 1 else null end) as areUnmarried, count(case
when maritalStatus = 'U' then 1 else null end) as setUnmarried from
account;
+-------+---------+--------------+--------------+
| Total | Married | areUnmarried | setUnmarried |
+-------+---------+--------------+--------------+
|  2024 |     470 |          777 |          777 |
+-------+---------+--------------+--------------+
```

This makes sense as the above statement creates a new tuple for each account holder that does not have maritalStatus = ‘M’. Therefore, we are essentially doubling the number of people that are unmarried thereby adding 777 rows to the 1247 making the new total 2024.