University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ___4_____

Date:June, 22_____

Group Number: __11_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Carson Lu	37798238	r3y3a	carson_lu@outlook.com
Jerry Fan	90420811	j0d7s	jerryxjfan@hotmail.com
Shekinah Titus	38809083	p5y2b	shekinahtitus7@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

2. SQL script can be found under Project/databaseSetup.sql

NOTE: the queries have NOT been included in the databaseSetup.sql file. There is contradicting information in the rubric and the milestone. The rubric asks for the table creations and inserts etc (which is done) but then the milestone pdf asks for queries? Queries that are used in this project are listed down below in part e.

3.

a. A short description of the final project, and what it accomplished.

The domain of the application is a database for various online services (both those that require and don't require subscriptions) and the focus is going to be on the data about the online services as well as the users of these online services. Our project focuses on Online Services and also the users and is targeted towards advertisers as they can see the most popular services or also the people that subscribe to the most subscriptions and focus their attention on advertising to them.

b. A description of how your final schema differed from the schema you turned in.

i. If the final schema differed, explain why. Note that turning in a final schema that's different from what you planned is fine, we just want to know what changed and why.

Schema mainly stayed the same, the only thing is that the decomposed tables are used. This is for simplicity as the user can view any table. Also, User was changed to UserData because sql*plus does not like it if we name the table User. Lastly, chars were turned into varchar2 and doubles were turned into number(*, 2)

c. A copy of the schema and screenshots that show what data is present in each relation after the SQL script from item #2 is run.

NOTE: Bolded ones are the ones that have been DECOMPOSED from the original, so bolded ones are no longer used and are replaced by those below
Photos are taken from IntelliJ after connecting to the database

ENTITIES:

`UserData(Email: varchar2[50], FirstName: varchar2[30], LastName: varchar2[30], Age: integer, Country: varchar2[60], PhoneNumber: integer)`

EMAIL	FIRSTNAME	LASTNAME	AGE	COUNTRY	PHONENUMBER
JohnDoe@gmail.com	John	Doe	25	United States	1234567890
JaneSmith@outlook.com	Jane	Smith	30	Canada	9876543210
MichaelJohnson@hotmail.com	Michael	Johnson	40	Australia	5678901234
EmilyBrown@yahoo.com	Emily	Brown	35	United Kingdom	9012345678
DavidWilson@webmail.com	David	Wilson	28	Germany	3456789012

Device(ModelNumber: integer, Brand: varchar2[20], Year: integer, OperatingSystem: varchar2[20], Platform: varchar2[20])
DeviceOS(OperatingSystem: varchar2(50), Platform varchar2(50)

OPERATINGSYSTEM	PLATFORM
iOS	Mobile
Windows	PC
Android	Mobile
macOS	PC
Linux	PC

DeviceModel(ModelNumber: integer, Brand: varchar2(50), Year: integer, OperatingSystem varchar2(50))

MODELNUMBER	BRAND	YEAR	OPERATINGSYSTEM
12345	Samsung	2022	Android
54321	Google	2023	Linux
67890	Apple	2021	iOs
98765	Microsoft	2020	macOS
23456	Sony	2022	Windows

Subscription(ServiceProvider : varchar2[20], BillingFrequency: varchar2[20], CostPerPayment: number(*, 2), YearlyCost: number(*, 2))
SubscriptionCosts(BillingFrequency: varchar2(50), CostPerPayment number(*, 2), YearlyCost: number(*, 2))

BILLINGFREQUENCY	COSTPERPAYMENT	YEARLYCOST
Monthly	9.99	119.88
Quarterly	24.99	99.96
Yearly	89.99	89.99
Monthly	12.99	155.88
Yearly	49.99	49.99

SubscriptionProvider(ServiceProvider: varchar2(50), BillingFrequency varchar2(50), CostPerPayment: number(*, 2))

SERVICEPROVIDER	BILLINGFREQUENCY	COSTPERPAYMENT
Amazon Prime	Yearly	119
Disney+	Yearly	79.99
Hulu	Monthly	11.99
Netflix	Monthly	9.99
Xbox Game Pass	Monthly	9.99

Review(Email: varchar2[50], ReviewID: integer, ServiceProvider: varchar2[20], Rating: integer)

ReviewDetails(ReviewID: integer, Rating: integer)

REVIEWID	RATING
1	4
2	5
3	3
4	4
5	2

ReviewOverview(Email: varchar2(50), ReviewID: integer, ServiceProvider: varchar(50))

EMAIL	REVIEWID	SERVICEPROVIDER
JohnDoe@gmail.com	1	Netflix
JaneSmith@outlook.com	2	Amazon Prime
MichaelJohnson@hotmail.com	3	Hulu
EmilyBrown@yahoo.com	4	Disney+
DavidWilson@webmail.com	5	Spotify

ServiceDataContains(ServiceProvider: varchar2[20], Version: varchar2[10], MonthlyUsers: integer, MonthlyRevenue: number(*, 2), MonthlyCosts: number(*, 2), MonthlyProfit: number(*, 2), Year: integer, Month: integer)

ServiceDataContainsCost(MonthlyCost: integer, MonthlyRevenue: integer, MonthlyProfit: integer)

MONTHLYCOST	MONTHLYREVENUE	MONTHLYPROFIT
1000	5000	4000
1500	6000	4500
1200	5500	4300
1800	7000	5200
900	4500	3600
1000	6000	5000

ServiceDataContainsProvider(ServiceProvider: varchar2(50),Version varchar2(10), MonthlyUsers integer,MonthlyCosts: integer, MonthlyRevenue: integer, Year: integer, Month: integer)

SERVICEPROVIDER	VERSION	MONTHLYUSERS	MONTHLYCOSTS	MONTHLYREVENUE	YEAR	MONTH
Netflix	1.0	100000	1000	5000	2023	1
Hulu	1.0	80000	1500	6000	2023	2
Amazon Prime	1.0	120000	1200	5500	2023	3
Disney+	1.0	90000	1800	7000	2023	4
Spotify	1.0	110000	900	4500	2023	5
Spotify	1.1	230000	1000	6000	2023	5

ServiceDataContainsVersion(Month: integer, Year: integer, Version: varchar2(10))

MONTH	YEAR	VERSION
1	2023	1.0
2	2023	1.1
3	2023	1.2
4	2023	1.3
5	2023	1.4

OnlineService(ServiceProvider: varchar2[20])

SERVICEPROVIDER
Amazon Prime
Deezer
Disney+
Epic Games
HBO Max
Hulu
Netflix
Origin
Pandora
PlayStation Now
Spotify
Steam
Tidal
Xbox Game Pass

MusicService(ServiceProvider: varchar2[20], MusicServiceName: varchar2[20])

SERVICEPROVIDER	MUSICSERVICENAME
Spotify	Spotify Music
Amazon Prime	Amazon Prime Music
Tidal	Tidal Music
Deezer	Deezer Music
Pandora	Pandora Music

MovieService(ServiceProvider: varchar2[20], MovieServiceName: varchar2[20])

SERVICEPROVIDER	MOVIESERVICENAME
Netflix	Netflix Movies
Amazon Prime	Amazon Prime Video
Hulu	Hulu Movies
Disney+	Disney+ Movies
HBO Max	HBO Max Movies

GameService(ServiceProvider: varchar2[20], GameServiceName: varchar2[20])

SERVICEPROVIDER	GAMESERVICENAME
Steam	Steam Games
Epic Games	Epic Games Store
Origin	Origin Games
Xbox Game Pass	Xbox Game Pass
PlayStation Now	PlayStation Now Games

Song(SongName: varchar2[20], SongYear: integer, SongGenre: varchar2[20])

SONGNAME	SONGYEAR	SONGGENRE
Song 1	2020	Pop
Song 2	2019	Rock
Song 3	2021	Hip Hop
Song 4	2018	Hip Hop
Song 5	2018	Hip Hop

Movie(MovieName: varchar2[20], MCompanyName: varchar2[20], MovieYear: integer, MovieGenre: varchar2[20])

MOVIENAME	MOVIEYEAR	MOVIEGENRE	MCOMPANYNAME
Ready Player One	2020	Sci-fi	Warner Bros.
Jurrasic World	2019	Action	Universal Pictures
Skyfall	2021	Drama	Columbia Pictures
Pokemon Detective Pikachu	2018	Adventure	Warner Bros.
Furious 7	2022	Science Fiction	Universal Pictures

VideoGame(GameName: varchar2[20], GameYear: integer, GameGenre: varchar2[20])

GAMENAME	GAMEYEAR	GAMEGENRE
Game 1	2020	Action
Game 2	2019	Adventure
Game 3	2021	RPG
Game 4	2018	Shooter
Game 5	2022	Sports

Artist(ArtistName: varchar2[20])

ARTISTNAME
Artist 1
Artist 2
Artist 3
Artist 4
Artist 5

MovieCompany(MCompanyName: varchar2[20])

MCOMPANYNAME
Columbia Pictures
Marvel Studios
Universal Pictures
Walt Disney Pictures
Warner Bros.

Developer(DevName: varchar2[20])

DEVNAME
Developer 1
Developer 2
Developer 3
Developer 4
Developer 5

RELATIONSHIPS:

Uses(Email: varchar2[50], ModelNumber: integer, Brand: integer)

EMAIL	MODELNUMBER	BRAND
DavidWilson@webmail.com	23456	Sony
EmilyBrown@yahoo.com	98765	Microsoft
JaneSmith@outlook.com	67890	Apple
JohnDoe@gmail.com	12345	Samsung
MichaelJohnson@hotmail.com	54321	Google

SubscribesTo(Email: varchar2[50], ServiceProvider: varchar2[20], BillingFrequency: varchar2[20], Price: number(*, 2))

EMAIL	SERVICEPROVIDER	BILLINGFREQUENCY	COSTPERPAYMENT	YEARLYCOST
JohnDoe@gmail.com	Netflix	Monthly	9.99	119.88
JohnDoe@gmail.com	Hulu	Monthly	11.99	143.88
JohnDoe@gmail.com	Amazon Prime	Yearly	119	119
JaneSmith@outlook.com	Hulu	Monthly	11.99	143.88
MichaelJohnson@hotmail.com	Amazon Prime	Yearly	119	119
EmilyBrown@yahoo.com	Disney+	Yearly	79.99	79.99
DavidWilson@webmail.com	Xbox Game Pass	Monthly	9.99	119.88

Accesses(ModelNumber: integer, Brand: varchar2[20], ServiceProvider: varchar2[20])

MODELNUMBER	BRAND	SERVICEPROVIDER
12345	Samsung	Netflix
23456	Sony	Spotify
54321	Google	Amazon Prime
67890	Apple	Hulu
98765	Microsoft	Disney+

Plays(ServiceProvider: varchar2[20], SongName: varchar2[20], SongYear: integer, NumPlays: integer)

SERVICEPROVIDER	SONGNAME	SONGYEAR	NUMPLAYS
Spotify	Song 1	2020	100
Amazon Prime	Song 2	2019	50
Tidal	Song 3	2021	75
Deezer	Song 4	2018	120
Pandora	Song 5	2018	90

Streams(ServiceProvider: varchar2[20], MovieName: varchar2[20], MovieYear: integer, NumStreams: integer)

SERVICEPROVIDER	MOVIENAME	MOVIEYEAR	NUMSTREAMS
Amazon Prime	Ready Player One	2020	200
Netflix	Jurrasic World	2019	150
Hulu	Skyfall	2021	180
Disney+	Pokemon Detective Pikachu	2018	220
HBO Max	Furious 7	2022	160

Provides(ServiceProvider: varchar2[20], GameName: varchar2[20], GameYear: integer, TimePlayed: integer)

SERVICEPROVIDER	GAMENAME	GAMEYEAR	TIMEPLAYED
Steam	Game 1	2020	10
Epic Games	Game 2	2019	8
Origin	Game 3	2021	12
Xbox Game Pass	Game 4	2018	15
PlayStation Now	Game 5	2022	20

`Writes(SongName: varchar2[20], SongYear: integer, ArtistName: varchar2[20])`

SONGNAME	SONGYEAR	ARTISTNAME
Song 1	2020	Artist 1
Song 2	2019	Artist 2
Song 3	2021	Artist 3
Song 4	2018	Artist 4
Song 5	2018	Artist 5

`Makes(GameName: varchar2[20], GameYear: integer, DevName: varchar2[20])`

GAMENAME	GAMEYEAR	DEVNAME
Game 1	2020	Developer 1
Game 2	2019	Developer 2
Game 3	2021	Developer 3
Game 4	2018	Developer 4
Game 5	2022	Developer 5

d. A list of all SQL queries used. For SQL query requirements, check the rubric listed on Canvas for Milestone 4.

e. Screenshots of the sample output of the queries using the GUI (for example, you can show what data is in your table before you run the query, and then show another screenshot after running the query, from some kind of GUI input like a button).

i. You need only to include screenshots for the required queries – if you implemented more than what was required, screenshots are not needed for those extra queries

NOTE PART d. AND e. ARE COMBINED AS THIS WILL BE EASIER TO SHOW WHICH QUERY LEADS TO WHICH RESULT

INSERT

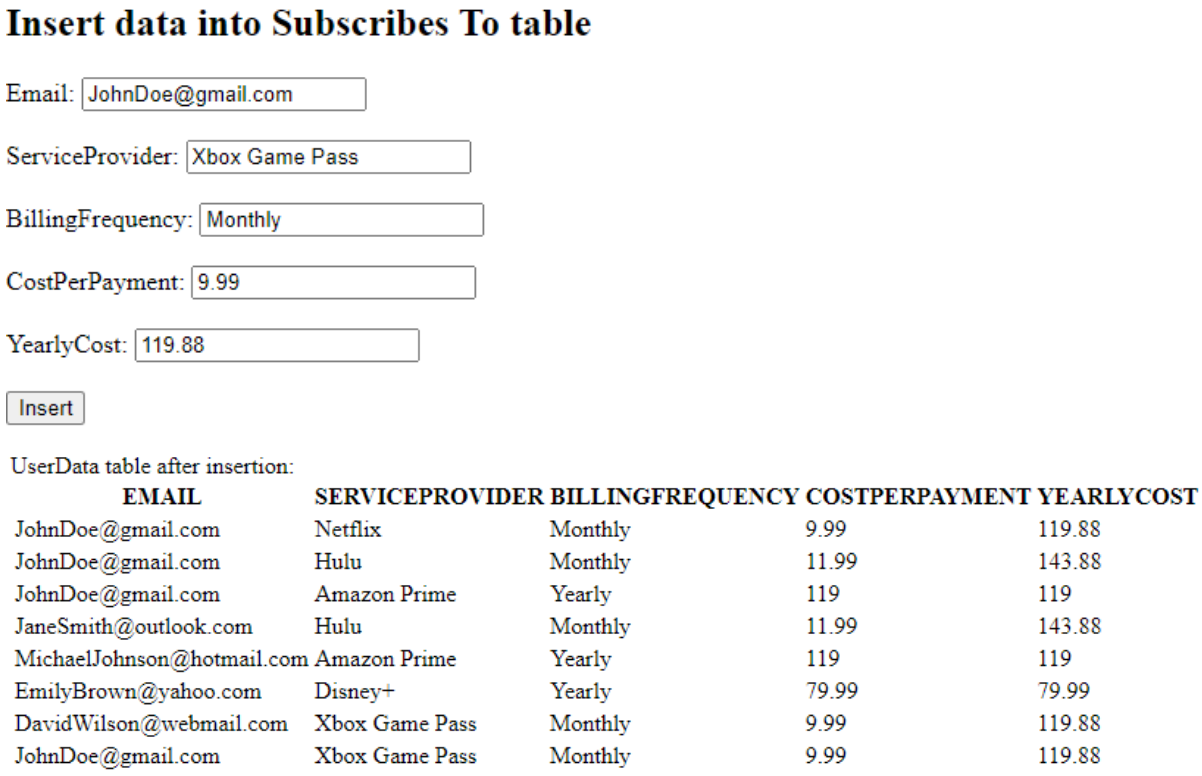
Query: (This is dynamic, here is the example one I used)

```
INSERT INTO SubscribesTo (Email, ServiceProvider,
BillingFrequency, CostPerPayment, YearlyCost)
VALUES ('JohnDoe@gmail.com', 'Xbox Game Pass', 'Monthly', 9.99,
198.88);
```

Before:

EMAIL	SERVICEPROVIDER	BILLINGFREQUENCY	COSTPERPAYMENT	YEARLYCOST
JohnDoe@gmail.com	Netflix	Monthly	9.99	119.88
JohnDoe@gmail.com	Hulu	Monthly	11.99	143.88
JohnDoe@gmail.com	Amazon Prime	Yearly	119	119
JaneSmith@outlook.com	Hulu	Monthly	11.99	143.88
MichaelJohnson@hotmail.com	Amazon Prime	Yearly	119	119
EmilyBrown@yahoo.com	Disney+	Yearly	79.99	79.99
DavidWilson@webmail.com	Xbox Game Pass	Monthly	9.99	119.88

After:



EMAIL	SERVICEPROVIDER	BILLINGFREQUENCY	COSTPERPAYMENT	YEARLYCOST
JohnDoe@gmail.com	Netflix	Monthly	9.99	119.88
JohnDoe@gmail.com	Hulu	Monthly	11.99	143.88
JohnDoe@gmail.com	Amazon Prime	Yearly	119	119
JaneSmith@outlook.com	Hulu	Monthly	11.99	143.88
MichaelJohnson@hotmail.com	Amazon Prime	Yearly	119	119
EmilyBrown@yahoo.com	Disney+	Yearly	79.99	79.99
DavidWilson@webmail.com	Xbox Game Pass	Monthly	9.99	119.88
JohnDoe@gmail.com	Xbox Game Pass	Monthly	9.99	119.88

Code:

```
1 usage  Carson
function handleInsertRequest()
{
    global $db_conn;

    //Getting the values from user and insert data into the table
    $tuple = array(
        ":email" => $_POST['emailInsert'],
        ":serviceProvider" => $_POST['serviceProviderInsert'],
        ":billingFrequency" => $_POST['billingFrequencyInsert'],
        ":costPerPayment" => $_POST['costPerPaymentInsert'],
        ":yearlyCost" => $_POST['yearlyCostInsert'],
    );

    $alltuples = array(
        $tuple
    );

    executeBoundSQL( cmdstr: "INSERT INTO SubscribesTo (Email, ServiceProvider, BillingFrequency, CostPerPayment,
     YearlyCost)
                    VALUES (:email, :serviceProvider, :billingFrequency, :costPerPayment, :yearlyCost)",
                     $alltuples);

    echo "UserData table after insertion:";
    printResultGeneral(executePlainSQL( cmdstr: "SELECT * FROM SubscribesTo"));
    // you need the wrap the old name and new name values with single quotations
    OCICommit($db_conn);
}
```

DELETE

Query:

```
DELETE
FROM UserData
WHERE Email = 'JohnDoe@gmail.com'
```

Before:

UserData table after delete:

EMAIL	FIRSTNAME	LASTNAME	AGE	COUNTRY	PHONENUMBER
JohnDoe@gmail.com	John	Doe	25	United States	1234567890
JaneSmith@outlook.com	Jane	Smith	30	Canada	9876543210
MichaelJohnson@hotmail.com	Michael	Johnson	40	Australia	5678901234
EmilyBrown@yahoo.com	Emily	Brown	35	United Kingdom	9012345678
DavidWilson@webmail.com	David	Wilson	28	Germany	3456789012

Potentially affected tables:

EMAIL	SERVICEPROVIDER	BILLINGFREQUENCY	COSTPERPAYMENT	YEARLYCOST
JohnDoe@gmail.com	Netflix	Monthly	9.99	119.88
JohnDoe@gmail.com	Hulu	Monthly	11.99	143.88
JohnDoe@gmail.com	Amazon Prime	Yearly	119	119
JaneSmith@outlook.com	Hulu	Monthly	11.99	143.88
MichaelJohnson@hotmail.com	Amazon Prime	Yearly	119	119
EmilyBrown@yahoo.com	Disney+	Yearly	79.99	79.99
DavidWilson@webmail.com	Xbox Game Pass	Monthly	9.99	119.88
JohnDoe@gmail.com	Xbox Game Pass	Monthly	9.99	119.88

EMAIL	MODELNUMBER	BRAND
DavidWilson@webmail.com	23456	Sony
EmilyBrown@yahoo.com	98765	Microsoft
JaneSmith@outlook.com	67890	Apple
JohnDoe@gmail.com	12345	Samsung
MichaelJohnson@hotmail.com	54321	Google

REVIEWID	EMAIL	SERVICEPROVIDER	RATING
1	JohnDoe@gmail.com	Netflix	4
2	JaneSmith@outlook.com	Amazon Prime	5
3	MichaelJohnson@hotmail.com	Hulu	3
4	EmilyBrown@yahoo.com	Disney+	4
5	DavidWilson@webmail.com	Spotify	2

After:

Delete data in UserData

Remove a user from the database (input their email)

Email to delete: JohnDoe@gmail.com

Delete

UserData table after delete:

EMAIL	FIRSTNAME	LASTNAME	AGE	COUNTRY	PHONENUMBER
JaneSmith@outlook.com	Jane	Smith	30	Canada	9876543210
MichaelJohnson@hotmail.com	Michael	Johnson	40	Australia	5678901234
EmilyBrown@yahoo.com	Emily	Brown	35	United Kingdom	9012345678
DavidWilson@webmail.com	David	Wilson	28	Germany	3456789012

Potentially affected tables:

EMAIL	SERVICEPROVIDER	BILLINGFREQUENCY	COSTPERPAYMENT	YEARLYCOST
JaneSmith@outlook.com	Hulu	Monthly	11.99	143.88
MichaelJohnson@hotmail.com	Amazon Prime	Yearly	119	119
EmilyBrown@yahoo.com	Disney+	Yearly	79.99	79.99
DavidWilson@webmail.com	Xbox Game Pass	Monthly	9.99	119.88

EMAIL	MODELNUMBER	BRAND
DavidWilson@webmail.com	23456	Sony
EmilyBrown@yahoo.com	98765	Microsoft
JaneSmith@outlook.com	67890	Apple
MichaelJohnson@hotmail.com	54321	Google

REVIEWID	EMAIL	SERVICEPROVIDER	RATING
2	JaneSmith@outlook.com	Amazon Prime	5
3	MichaelJohnson@hotmail.com	Hulu	3
4	EmilyBrown@yahoo.com	Disney+	4
5	DavidWilson@webmail.com	Spotify	2

Code:

```
1 usage   Carson *
function handleDeleteRequest()
{
    global $db_conn;
    $tuple = array(
        ":email" => $_POST['emailDelete']
    );

    $alltuples = array(
        $tuple
    );
    executeBoundSQL( cmdstr: "
                DELETE
                FROM UserData
                WHERE Email = :email
            ", $alltuples);
    OCICommit($db_conn);
    echo "<h2>UserData table after delete:</h2>";
    printResultGeneral(executePlainSQL( cmdstr: "SELECT * FROM UserData"));
    echo "<h2>Potentially affected tables:</h2>";
    printResultGeneral(executePlainSQL( cmdstr: "SELECT * FROM SubscribesTo"));
    printResultGeneral(executePlainSQL( cmdstr: "SELECT * FROM Uses"));
    printResultGeneral(executePlainSQL( cmdstr: "SELECT * FROM ReviewOverview NATURAL JOIN ReviewDetails"));
}
```

UPDATE

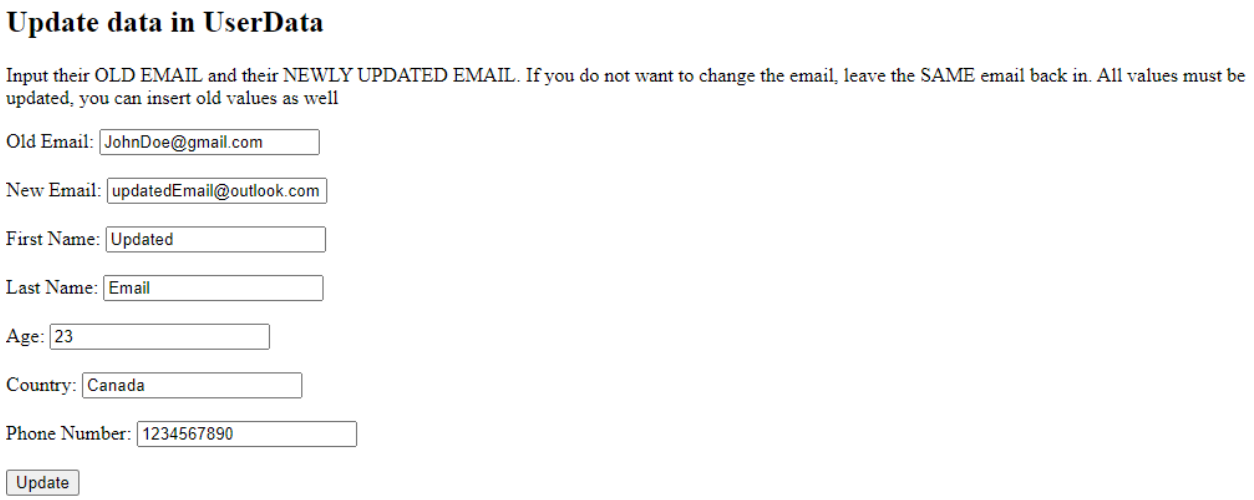
Query:

```
UPDATE UserData
SET Email='updatedEmail@outlook.com', FirstName='Updated',
LastName='Email', Age=23, Country='Canada', PhoneNumber=
1234567890
WHERE Email=JohnDoe@gmail.com
```

Before:

EMAIL	FIRSTNAME	LASTNAME	AGE	COUNTRY	PHONENUMBER
JohnDoe@gmail.com	John	Doe	25	United States	1234567890
JaneSmith@outlook.com	Jane	Smith	30	Canada	9876543210
MichaelJohnson@hotmail.com	Michael	Johnson	40	Australia	5678901234
EmilyBrown@yahoo.com	Emily	Brown	35	United Kingdom	9012345678
DavidWilson@webmail.com	David	Wilson	28	Germany	3456789012

After:



UserData table after update:

EMAIL	FIRSTNAME	LASTNAME	AGE	COUNTRY	PHONENUMBER
updatedEmail@outlook.com	Updated	Email	23	Canada	1234567890
JaneSmith@outlook.com	Jane	Smith	30	Canada	9876543210
MichaelJohnson@hotmail.com	Michael	Johnson	40	Australia	5678901234
EmilyBrown@yahoo.com	Emily	Brown	35	United Kingdom	9012345678
DavidWilson@webmail.com	David	Wilson	28	Germany	3456789012

Code:

```php
1 usage  Carson *
function handleUpdateRequest()
{
    global $db_conn;

    $tuple = array(
        ":old_email" => $_POST['oldEmailUpdate'],
        ":new_email" => $_POST['newEmailUpdate'],
        ":new_fname" => $_POST['fnameUpdate'],
        ":new_lname" => $_POST['lnameUpdate'],
        ":new_age" => $_POST['ageUpdate'],
        ":new_country" => $_POST['countryUpdate'],
        ":new_phoneNumber" => $_POST['phoneNumberUpdate']
    );

    $alltuples = array(
        $tuple
    );

    executeBoundSQL( cmdstr: "UPDATE UserData SET Email=:new_email, FirstName=:new_fname,
     LastName=:new_lname, Age=:new_age, Country=:new_country, PhoneNumber=:new_phoneNumber WHERE
     Email=:old_email", $alltuples);

    echo "<h2>UserData table after update:</h2>";
    printResultGeneral(executePlainSQL( cmdstr: "SELECT Email, FirstName, LastName, Age, Country,
     PhoneNumber FROM UserData"));

    OCICommit($db_conn);
}
```

SELECTION

Query:

```
SELECT *
FROM UserData
```

```
WHERE Email = ‘MichaleJohnson@hotmail.com
```

Before:

Pick what table you would like to view and

Separate each new column should be separated by a comma. Leave no spaces between.

If you would like to get every column, put a * in the "columns" textbox

For the "where" textbox, you should use inequalities, that is, it should look something like column = 'value' or column > number

Table Name: UserData

Columns: *

Where: Email = 'MichaelJohnson@h

Select

After:

EMAIL	FIRSTNAME	LASTNAME	AGE	COUNTRY	PHONENUMBER
MichaelJohnson@hotmail.com	Michael	Johnson	40	Australia	5678901234

Code:

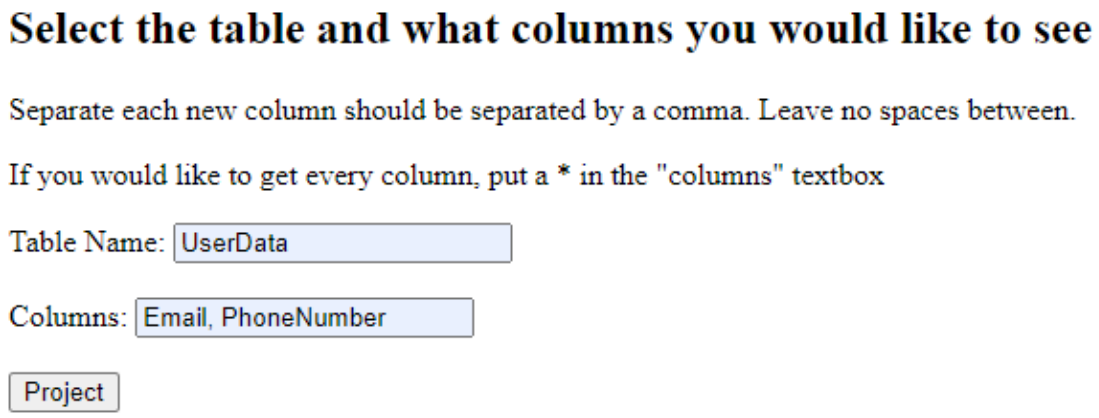
```php
1 usage  Carson
function handleSelectionRequest()
{
    global $db_conn;
    $table = $_GET['tableSelect'];
    $columns = $_GET['columnSelect'];
    $where = $_GET['whereSelect'];

    $result = executePlainSQL( cmdstr: "SELECT $columns FROM $table WHERE $where");

    if ($result) {
        printResultGeneral($result);
    } else {
        echo "Invalid, no result";
    }
}
```

PROJECTION

Query:

```
SELECT *
FROM UserData
```

Before:



After:

EMAIL	PHONENUMBER
updatedEmail@outlook.com	1234567890
JaneSmith@outlook.com	9876543210
MichaelJohnson@hotmail.com	5678901234
EmilyBrown@yahoo.com	9012345678
DavidWilson@webmail.com	3456789012

Code:

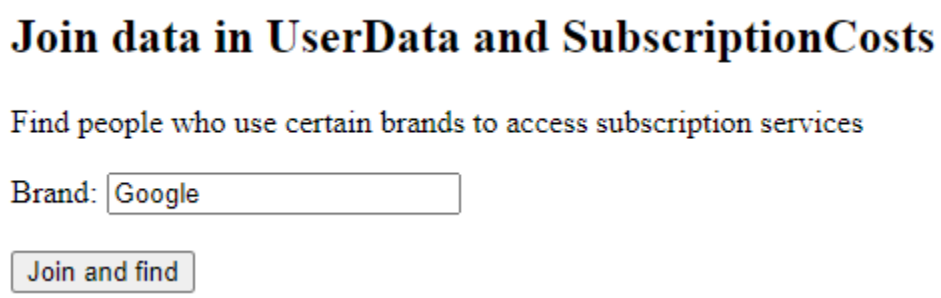
```php
1 usage  Carson
function handleProjectRequest()
{
    global $db_conn;
    $table = $_GET['tableProject'];
    $columns = $_GET['columnProject'];

    $result = executePlainSQL( cmdstr: "SELECT $columns FROM $table");

    if ($result) {
        printResultGeneral($result);
    } else {
        echo "Invalid, no result";
    }
}
```

JOIN

Query:

```
SELECT *
FROM Uses U, DeviceOS DOS, DeviceModel DM
WHERE DOS.OperatingSystem = DM.OperatingSystem AND
U.ModelNumber = DM.ModelNumber AND U.ModelNumber =
DM.ModelNumber
```

Before:



After:

Unfiltered table:

EMAIL	MODELNUMBER	BRAND	OPERATINGSYSTEM	PLATFORM	YEAR
DavidWilson@webmail.com	23456	Sony	Windows	PC	2022
JohnDoe@gmail.com	12345	Samsung	Android	Mobile	2022
EmilyBrown@yahoo.com	98765	Microsoft	macOS	PC	2020
MichaelJohnson@hotmail.com	54321	Google	Linux	PC	2023

Table with only items from brand: Google

EMAIL	MODELNUMBER	BRAND	OPERATINGSYSTEM	PLATFORM	YEAR
MichaelJohnson@hotmail.com	54321	Google	Linux	PC	2023

Code:

```php
1 usage  Carson
function handleJoinRequest()
{
    global $db_conn;
    $brand = $_GET['brand'];
    $result = executePlainSQL( cmdstr: "SELECT * FROM Uses U, DeviceOS DOS, DeviceModel DM WHERE DOS
     .OperatingSystem = DM.OperatingSystem AND U.ModelNumber = DM.ModelNumber AND U.ModelNumber =
     DM.ModelNumber AND U.Brand = '" . $brand . "'");

    echo "<h3>Unfiltered table:</h3>";
    printResultGeneral(executePlainSQL( cmdstr: "SELECT * FROM Uses U, DeviceOS DOS, DeviceModel DM WHERE
     DOS.OperatingSystem = DM.OperatingSystem AND U.ModelNumber = DM.ModelNumber AND U.ModelNumber =
     DM.ModelNumber"));

    echo "<h3>Table with only items from brand: $brand</h3>";
    printResultGeneral($result);
}
```

AGGREGATION WITH GROUP BY

Query:

```
SELECT ServiceProvider, Year, AVG(MonthlyRevenue)
FROM ServiceDataContainsProvider
GROUP BY ServiceProvider, Year
```

Before:

Find the average monthly revenue of companies by each year

Get Data

After:

SERVICEPROVIDER	YEAR	AVG(MONTHLYREVENUE)
Hulu	2023	6000
Spotify	2023	5250
Netflix	2023	5000
Disney+	2023	7000
Amazon Prime	2023	5500

Code:

```php
1 usage  Carson
function handleGroupByRequest()
{
    global $db_conn;
    $result = executePlainSQL( cmdstr: "SELECT ServiceProvider, Year, AVG(MonthlyRevenue)
                                        FROM ServiceDataContainsProvider
                                        GROUP BY ServiceProvider, Year");

    if (!$result) {
        echo "Error executing the query.";
        return;
    }

    printResultGeneral($result);
}
```

AGGREGATION WITH HAVING

Query:

```
SELECT u.Email, COUNT(*)
FROM UserData u, SubscribesTo s
WHERE u.Email = s.Email
GROUP BY u.Email
HAVING COUNT(*) > 2
```

Before:

Find all emails that are subscribed to more than 2 subscriptions

Get data

After:

EMAIL	COUNT(*)
JohnDoe@gmail.com	3

Code:

```
1 usage   Carson
function handleHavingRequest()
{
    global $db_conn;
    $result = executePlainSQL( cmdstr: "SELECT u.Email, COUNT(*)
                                        FROM UserData u, SubscribesTo s
                                        WHERE u.Email = s.Email
                                        GROUP BY u.Email
                                        HAVING COUNT(*) > 2");
    printResultGeneral($result);
}
```

NESTED AGGREGATION WITH GROUP BY

Query:

```
SELECT ServiceProvider, AVG(MonthlyRevenue)
FROM ServiceDataContainsProvider
GROUP BY ServiceProvider
HAVING AVG(MonthlyRevenue) > (SELECT AVG(MonthlyRevenue) FROM
ServiceDataContainsProvider)
```

Before:

Find all subscriptions with a higher than average monthly revenue

Get Data

After:

SERVICEPROVIDER	AVG(MONTHLYREVENUE)
Disney+	7000
Hulu	6000

Code:

```
1 usage   Carson *
function handleNestedGroupRequest()
{
    global $db_conn;
    $result = executePlainSQL( cmdstr: "SELECT ServiceProvider, AVG(MonthlyRevenue)
                                        FROM ServiceDataContainsProvider
                                        GROUP BY ServiceProvider
                                        HAVING AVG(MonthlyRevenue) > (SELECT AVG(MonthlyRevenue) FROM
                                         ServiceDataContainsProvider)");
    printResultGeneral($result);

}
```

DIVISION

Query:

Before:

Get all online services that have been used by all device platforms (Mobile and PC)

Get division data

After:

SERVICEPROVIDER
Amazon Prime
Disney+
Hulu
Netflix
Spotify

Code:

```
SELECT *
FROM OnlineService O
WHERE NOT EXISTS ((SELECT Platform FROM DeviceOS DOS NATURAL
JOIN DeviceModel DM)
MINUS (SELECT DOS2.Platform FROM Accesses A, DeviceOS DOS2,
DeviceModel DM2 WHERE DM2.ModelNumber = A.ModelNumber AND
DM2.Brand = A.Brand AND O.ServiceProvider =
A.ServiceProvider))");
```

```
1 usage   Carson
function handleDivisionRequest()
{
    global $db_conn;
    $result = executePlainSQL( cmdstr: "SELECT * FROM OnlineService O
                                        WHERE NOT EXISTS
                                            ((SELECT Platform FROM DeviceOS DOS NATURAL JOIN DeviceModel
                                              DM)
                                            MINUS (SELECT DOS2.Platform
                                                   FROM Accesses A, DeviceOS DOS2, DeviceModel DM2
                                                   WHERE DM2.ModelNumber = A.ModelNumber
                                                     AND DM2.Brand = A.Brand
                                                     AND O.ServiceProvider = A.ServiceProvider))");
    printResultGeneral($result);
}
```