Database Project

CSI2132 Section A

Professor: Dr. Paula Branco

Students: Christopher Aris
300031710, caris015@uottawa.ca
Lev C. Guzman Aparicio
300038033, lguzm038@uottawa.ca

April 7, 2020

1 Languages Used

To build the database project, the team used a combination of Python and SQL PostgreSQL flavour.

Python was chosen in order to ease work and concentrate on the main task of this project: learn SQL and how to manage a database.

The team decided to go with a CLI interface because it was the cleanest solution; and made it so we could focus on other aspects of the project, however the main reason was robustness. A CLI can be a very powerful tool, a good example is Heroku, and even though ours is nowhere nearly as sophisticated, with a few lines of code we were able to create and read information remotely.

The following DDLs were used in order to create our database:

- CREATE TABLE ... (...);
- ALTER TABLE (...);
- ALTER COLUMN ... TYPE ...;
- RENAME ... TO ...;
- ADD CONSTRAINT ... FOREIGN KEY (...) REFERENCES ...(...) ON DELETE ... ON UPDATE ...;
- ADD COLUMN ...;
- ADD CONSTRAINT ... CHECK (...);
- ADD CONSTRAINT ... PRIMARY KEY (...);
- ADD CONSTRAINT ... NOT NULL(...);
- DROP CONSTRAINT;
- DROP COLUMN;

2 Queries

Ten queries were requested for our database. We present their code and the corresponding results

1. ```
   -- Query #1
   SELECT user_info.first_name, user_info.last_name, properties_info.type,
      properties_info.pricing_id, rental_agreement.start_date,
      properties_info.location, payment_info.type_of_payment, payment_info
      .payment_status
   FROM users.guest, users.user_info, properties.properties_info,
      properties.rental_agreement, transactions.payment_info
   WHERE rental_agreement.guest_signee_id = guest.guest_id AND guest.
      guest_id = user_info.account_id
   ORDER BY payment_info.type_of_payment ASC, rental_agreement.start_date
      DESC
   ```

Data Output | Explain | Messages | Notifications

	first_name character varying (20)	last_name character varying (20)	type character varying (20)	pricing_id integer	start_date date	location character varying (20)	type_of_payment character varying (20)	payment_status character varying (20)
601	Cesar	Mika	house	895	2000-12-23	Tlaxcala	Cash	Accepted
602	Cesar	Mika	apartment	840	2000-12-23	Texas	Cash	Accepted
603	Cesar	Mika	villa	730	2000-12-23	Baja California	Cash	Accepted
604	Cesar	Mika	villa	764	2000-12-23	Ontario	Cash	Not Accepted
605	Cesar	Mika	room	872	2000-12-23	Quebec	Cash	Not Accepted
606	Cesar	Mika	condo	841	2000-12-23	Tlaxcala	Cash	Not Accepted
607	Cesar	Mika	apartment	799	2000-12-23	Tamaulipas	Cash	Not Accepted
608	Cesar	Mika	condo	813	2000-12-23	Baja California	Cash	Not Accepted
609	Cesar	Mika	mansion	836	2000-12-23	California	Cash	Not Accepted
610	Cesar	Mika	room	750	2000-12-23	Texas	Cash	Not Accepted
611	Cesar	Mika	basement	803	2000-12-23	Texas	Cash	Not Accepted
612	Cesar	Mika	villa	775	2000-12-23	Alberta	Cash	Not Accepted
613	Cesar	Mika	basement	768	2000-12-23	Alberta	Cash	Not Accepted
614	Cesar	Mika	house	833	2000-12-23	Ontario	Cash	Not Accepted
615	Cesar	Mika	villa	899	2000-12-23	Alberta	Cash	Not Accepted
616	Cesar	Mika	mansion	821	2000-12-23	Chihuahua	Cash	Not Accepted
617	Cesar	Mika	apartment	847	2000-12-23	Mexico	Cash	Not Accepted
618	Cesar	Mika	villa	846	2000-12-23	New Jersey	Cash	Not Accepted
619	Cesar	Mika	room	890	2000-12-23	Alberta	Cash	Not Accepted
620	Cesar	Mika	apartment	817	2000-12-23	Alberta	Cash	Not Accepted

Figure 1: Output of the 1st Query
   ```

2.
```
-- Query #2
CREATE VIEW users.GuestListView AS
SELECT guest.guest_id, user_info.account_id, user_info.first_name,
    user_info.last_name, user_info.email, user_info.country FROM users.
    user_info
JOIN users.guest ON user_info.account_id = guest.guest_id
ORDER BY guest.guest_id
```

Data Output | Explain | Messages | Notifications

	guest_id integer	account_id integer	first_name character varying (20)	last_name character varying (20)	email character varying	country character varying (20)
1	0	0	Cesar	Sekhon	queen798@service.com	USSR
2	1	1	Ehecatl	Choolhon	queen037@service.com	Canada
3	2	2	Chris	Guzman	sars690@service.com	Russia
4	3	3	Andrew	Mika	quetzal975@service.c...	United States
5	4	4	Chris	Mika	queen937@service.com	Spain
6	5	5	Cesar	Sekhon	sars913@service.com	USSR
7	6	6	Anne	D'souza	quarantinegal957@ser...	Russia
8	7	7	Ehecatl	D'souza	elote536@service.com	Colombia
9	8	8	Savy	Hernandez	chachalaca273@servic...	Japan
10	9	9	Jason	Li	quetzal081@service.c...	Brazil
11	10	10	Sandy	D'souza	chachalaca431@servic...	Ukraine
12	11	11	Ehecatl	Eufracio	elote684@service.com	Brazil
13	12	12	Lev	Wedia	elote254@service.com	Colombia
14	13	13	Omer	Guzman	user276@service.com	Spain
15	14	14	Rebeca	Wu	tlatoani394@service.c...	Indonesia
16	15	15	David	Abubaker	belmont610@service.c...	Belorussia
17	16	16	Sam	Wedia	queen097@service.com	Indonesia
18	17	17	Melissa	Sekhon	mersh1n1401@service...	Canada
19	18	18	Andrew	Bundhoo	quarantineboi971@ser...	Colombia
20	19	19	Savy	Eufracio	postgres976@service....	Spain
21	20	20	Anne	Mika	alucard721@service.c...	Canada
22	21	21	David	Abubaker	elote918@service.com	China
23	22	22	Camron	Mika	h5n1250@service.com	Colombia
24	23	23	Cesar	Mika	sars941@service.com	United States
25	24	24	Lev	Branco	quarantinegal512@ser...	Belorussia

Figure 2: Output of the 2nd Query

3.
```
-- Query #3
SELECT * FROM properties.properties_info, users.guest, properties.
    rental_agreement, users.user_info
WHERE properties_info.property_id = rental_agreement.property_id AND
    user_info.account_id = guest.guest_id
ORDER BY properties_info.pricing_id LIMIT 1  -- returns the top entry
    which is the lowest number since by default items are arranged in
    ascending order
```

Data Output | Explain | Messages | Notifications

	property_id integer	type character varying (20)	number_of_rooms integer	owner_id integer	available_date date	pricing_id integer	location character varying (20)	guest_id integer	account_id integer	agreement_id integer	property_id integer	start_date date	end_date date
1	542	basement	10	32	2000-06-01	701	Mexico	0	0	5030	542	1994-11-06	2018-05-17

Figure 3: Output of the 3nd Query

4.
```
-- Query #4
SELECT properties_info.property_id, properties_info.type,
   properties_info.number_of_rooms, properties_info.location,
   review_info.number_of_stars
FROM properties.properties_info, properties.rental_agreement, management
   .review_info, users.guest
WHERE rental_agreement.guest_signee_id = guest.guest_id
GROUP BY properties_info.property_id, review_info.number_of_stars
ORDER BY review_info.number_of_stars
```

Data Output | Explain | Messages | Notifications

	property_id integer	type character varying (20)	number_of_rooms integer	location character varying (20)	number_of_stars integer
186	557	apartment	2	New Mexico	1
187	664	condo	9	Mexico	1
188	522	basement	9	Tlaxcala	1
189	547	apartment	10	California	1
190	556	condo	7	Nunavut	1
191	597	house	6	Tlaxcala	1
192	632	room	9	New Jersey	1
193	520	basement	9	Texas	1
194	535	room	1	Nunavut	1
195	533	apartment	7	New Jersey	1
196	643	house	4	Oaxaca	1
197	568	basement	2	Chihuahua	1
198	671	house	9	Ciudad de Mexico	1
199	500	villa	10	Ontario	1
200	544	condo	7	Tlaxcala	1
201	669	apartment	7	Baja California	1
202	663	villa	2	California	2
203	503	mansion	1	Oaxaca	2
204	646	room	10	Nunavut	2
205	691	villa	6	Alberta	2
206	639	house	6	Mexico	2
207	659	condo	1	Alberta	2
208	562	condo	1	Sonora	2

Figure 4: Output of the 4th Query.

5.
```
--Query #5
SELECT properties_info.property_id, type, number_of_rooms, owner_id,
    available_date, properties_info.pricing_id, location
FROM  properties_info, rental_agreement
WHERE rental_agreement.property_id = null; --a null value in
    rental_agreement says it is not rented
```

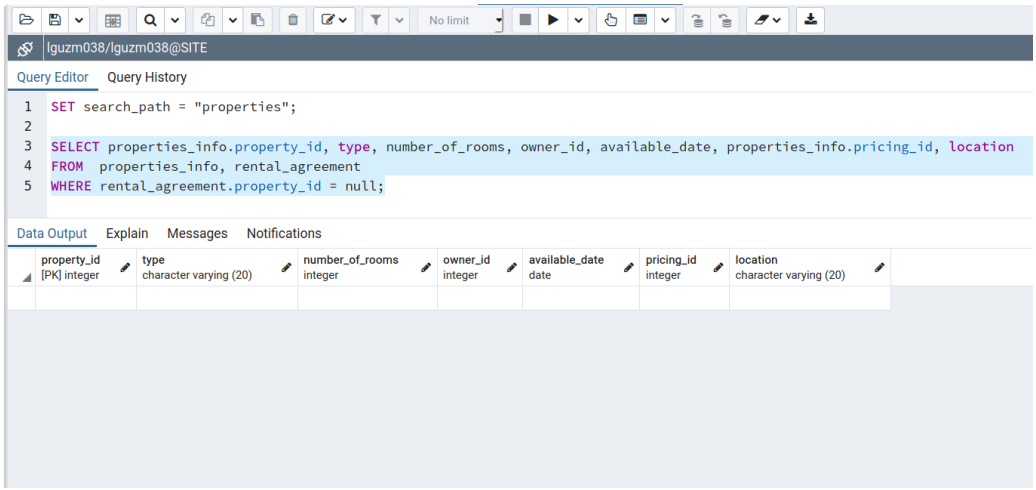

Figure 5: Output of the 5th Query. Note that it is empty since we inserted no null values

6.
```
--Query #6
SELECT properties_info.property_id, properties_info.type,
    properties_info.number_of_rooms, properties_info.location
FROM properties.properties_info, properties.rental_agreement
WHERE rental_agreement.start_date = '1991-03-10'
```

Data Output | Explain | Messages | Notifications

	property_id integer	type character varying (20)	number_of_rooms integer	location character varying (20)	start_date date
1	500	villa	10	Ontario	1991-03-10
2	501	basement	8	Tlaxcala	1991-03-10
3	502	condo	8	Sonora	1991-03-10
4	503	mansion	1	Oaxaca	1991-03-10
5	504	mansion	2	California	1991-03-10
6	505	house	9	Tlaxcala	1991-03-10
7	506	basement	1	Chihuahua	1991-03-10
8	507	basement	9	Oaxaca	1991-03-10
9	508	room	8	Sonora	1991-03-10
10	509	room	9	Quebec	1991-03-10
11	510	condo	6	Baja California	1991-03-10
12	511	basement	9	Quebec	1991-03-10
13	512	mansion	8	Mexico	1991-03-10
14	513	basement	4	Sonora	1991-03-10
15	514	house	3	Texas	1991-03-10
16	515	condo	8	Nova Scotia	1991-03-10
17	516	mansion	7	California	1991-03-10
18	517	basement	1	New Mexico	1991-03-10
19	518	house	3	California	1991-03-10
20	519	room	4	Oaxaca	1991-03-10
21	520	basement	9	Texas	1991-03-10
22	521	basement	5	California	1991-03-10
23	522	basement	9	Tlaxcala	1991-03-10
24	523	basement	1	Tlaxcala	1991-03-10
25	524	mansion	7	Sonora	1991-03-10
26	525	apartment	1	Alberta	1991-03-10

Figure 6: Output of the 6th Query.

7.
```
--Query #7
SELECT employee_info.employee_id, employee_info.first_name,
   employee_info.last_name, employee_info.works_for_branch_id,
   branch_info.branch_name, employee_info.salary
FROM management.employee_info, management.branch_info
WHERE employee_info.salary >= 1500  --changed from 15,000 because no one
    makes that much money in our company!
ORDER BY employee_info.employee_id, employee_info.manages_branch_id
```

Data Output | Explain | Messages | Notifications

	property_id integer	type character varying (20)	number_of_rooms integer	location character varying (20)	start_date date
1	500	villa	10	Ontario	1991-03-10
2	501	basement	8	Tlaxcala	1991-03-10
3	502	condo	8	Sonora	1991-03-10
4	503	mansion	1	Oaxaca	1991-03-10
5	504	mansion	2	California	1991-03-10
6	505	house	9	Tlaxcala	1991-03-10
7	506	basement	1	Chihuahua	1991-03-10
8	507	basement	9	Oaxaca	1991-03-10
9	508	room	8	Sonora	1991-03-10
10	509	room	9	Quebec	1991-03-10
11	510	condo	6	Baja California	1991-03-10
12	511	basement	9	Quebec	1991-03-10
13	512	mansion	8	Mexico	1991-03-10
14	513	basement	4	Sonora	1991-03-10
15	514	house	3	Texas	1991-03-10
16	515	condo	8	Nova Scotia	1991-03-10
17	516	mansion	7	California	1991-03-10
18	517	basement	1	New Mexico	1991-03-10
19	518	house	3	California	1991-03-10
20	519	room	4	Oaxaca	1991-03-10
21	520	basement	9	Texas	1991-03-10
22	521	basement	5	California	1991-03-10
23	522	basement	9	Tlaxcala	1991-03-10
24	523	basement	1	Tlaxcala	1991-03-10
25	524	mansion	7	Sonora	1991-03-10
26	525	apartment	1	Alberta	1991-03-10

Figure 7: Output of the 7th Query.

8.
```
--Query #8
SELECT properties_info.type, user_info.first_name, user_info.last_name,
    properties_info.location, rental_agreement.pricing_id, payment_info
   .type_of_payment
FROM users.user_info, properties.properties_info, properties.
   rental_agreement, users.host, transactions.payment_info
WHERE properties_info.owner_id = host.host_id AND host.host_id =
   user_info.account_id   -- to specify for a specific user you would
   also add another AND with host_id being a specific integer
```

Data Output | Explain | Messages | Notifications

	type character varying (20)	first_name character varying (20)	last_name character varying (20)	location character varying (20)	pricing_id integer	type_of_payment character varying (20)
185	villa	David	Aparicio	Ontario	800	Cash
186	villa	David	Aparicio	Ontario	812	Cash
187	villa	David	Aparicio	Ontario	840	Cash
188	villa	David	Aparicio	Ontario	762	Cash
189	villa	David	Aparicio	Ontario	827	Cash
190	villa	David	Aparicio	Ontario	808	Cash
191	villa	David	Aparicio	Ontario	730	Cash
192	villa	David	Aparicio	Ontario	842	Cash
193	villa	David	Aparicio	Ontario	756	Cash
194	villa	David	Aparicio	Ontario	812	Cash
195	villa	David	Aparicio	Ontario	755	Cash
196	villa	David	Aparicio	Ontario	705	Cash
197	villa	David	Aparicio	Ontario	884	Cash
198	villa	David	Aparicio	Ontario	855	Cash
199	villa	David	Aparicio	Ontario	797	Cash
200	villa	David	Aparicio	Ontario	765	Cash
201	basement	Melissa	Sekhon	Tlaxcala	824	Cash
202	basement	Melissa	Sekhon	Tlaxcala	751	Cash
203	basement	Melissa	Sekhon	Tlaxcala	823	Cash
204	basement	Melissa	Sekhon	Tlaxcala	760	Cash
205	basement	Melissa	Sekhon	Tlaxcala	881	Cash

Figure 8: Output of the 8th Query.

9.
```
--Query #9
UPDATE users.user_info
SET phone_number = '6132495439'
FROM users.guest
WHERE user_info.account_id = guest.guest_id AND user_info.first_name = '
   Jason' AND user_info.last_name = 'Wu'
```

Figure 9: Output of the 9th Query.

10.

```
--Query #10
CREATE FUNCTION FirstNameFirst(firstName char(20), lastName char(20))
    RETURNS char(50) AS $$
  SELECT CONCAT (firstName, ' ', lastName) AS FullName;

  $$ LANGUAGE SQL;
```

Figure 10: Output of the 10th Query.

3 How to Install

To set up the database we must follow these steps in order:

- Restore the provided backup file in order to access the schema and tables.
- In the *Data* folder, execute the provided python script **data_insertor.py** with the following command: `python3 data_insertor.py`

  Remark. *You will need to modify the connection fields and enter your own credentials in line 18 of data_insertor in order to ensure proper functioning.*

- Now de database is populated.

To install the CLI tool it is **important** that the guide is followed strictly in order to make sure that no errors present themselves.

- It is *strongly* advised that the program is ran in a virtual environment
- It is **even more so** advised that this installation is executed using the latest version of Python, but any Python 3.X version will suffice.

  Remark. *Please **do not use** Python 2.X binaries because there **WILL** be errors.*

- Once in a virtual environment run the command: `pip install --editable .`

  What this is essentially telling Python is to install the package in the current directory (which you guessed it, is our CLI Tool), the editable option will link the package to the directory location and mitigate any nasty import errors.

- If on a UNIX system you might have to run the following instead:
  `pip3 install --editable .`

  This specifies to your system to select a Python 3.X version if a Python 2.X is also installed (which is usally the case with Linux distros)

- Once installed verify installation by checking for the package 'travelCLI v1.0' To do this, type `pip list` or `pip3 list` (if on a UNIX system)

- To finally start using the CLI Tool, the binding keyword is `controller` so if I were to run the check command I would do like so: `$ controller check`

  Remark. *Note that you will need to edit the configuration file titled 'database.ini' to change the credentials to appropriate ones; in order to connect to the proper database source.*

4 Final Thoughts

From the project we learned a lot about how databases work. We were able to create our own database with different own-designed schema. From the project we learned a lot about how a DBMS can interact with a server, or a user.

If we had more time on our hands we would have liked to add more features to our CLI with main being making use of prompts to help with interactivity and automating the sign-in process. We would have also liked to polish our outputs by making use of formatted tables.
In summation however we feel that with the time and responsibilities we had we executed to our best effort and are pleased with our current result