

Test No	Test Description	Result
1	Database Connection: ensure python connection to db allows queries	PASSED
2	SQL query for primary key works. Check that sql for "TAB-000", works to return the next value	PASSED
3	INSERT/UPDATE/DELETE in Single Table. Ensure that table(s) can be edited (following any constraints)	PASSED
4	INSERT into Db breaking Key contraints. Force INSERT error to confirm contraint functions as intended	PASSED
5	INSERT into Db breaking UNQIUE contraints. Force INSERT error to confirm contraint functions as intended	PASSED

Test No	Test Description	Result
6	Trigger Creation & execution. Confirm creation without error	PASSED
7	Trigger INSERT/DELETE. Confirm trigger creation & required outputs without error (NEW only for INSERT, and OLD for DELETE)	PASSED
8	Trigger UPDATE. Confirm trigger creation & required outputs without error (NEW and OLD data pulled)	PASSED
9	VIEW creation & execution. Ensure data is returned as expected for created VIEW	PASSED

Test No	Test Description	Result	Test Data/Screenshots
1	<p>Database Connection:</p> <p>Ensure python connection to db allows queries. Setup pytest to connect to the database & run a confirmed script to assert that data has been returned.</p>	PASSED	Below

```
● (.venv) PS C:\Users\eliot\Documents\COMS19> pytest -v tests/test_utility_functions.py::test_database_connection_pass
=====
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- c:\Users\eliot\Documents\COMS19\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\eliot\Documents\COMS19
collected 1 item

tests/test_utility_functions.py::test_database_connection_pass PASSED [100%]

===== 1 passed in 0.81s =====
```

Full test script:

```
def test_database_connection_pass():
    """
    confirm that database connection works & passed back in function
    """

    import database as db

    settings = get_settings_data()

    db_path = os.path.join(
        settings["database_settings"]["database_path"],
        settings["database_settings"]["database"]
    )

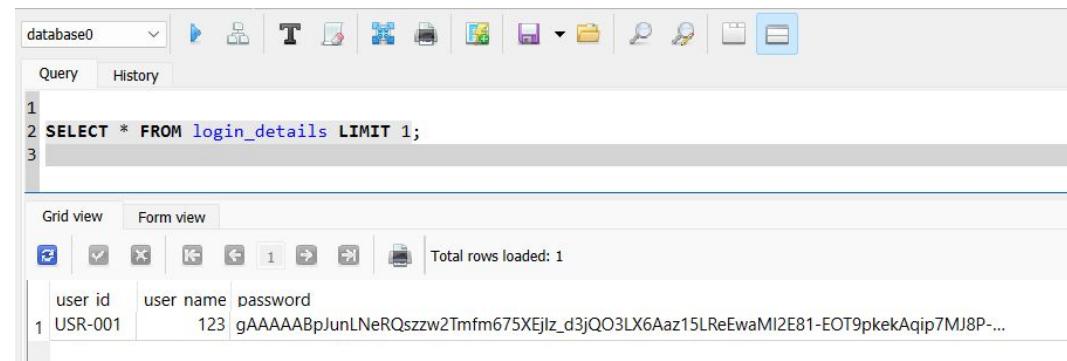
    # Create a connection
    conn = db.Database(db_path)

    # Test connection
    test_sql = load_sql_file("database_connection_check.sql")
    result = conn.query(test_sql, {})

    conn.close()

    assert result != None
```

Confirm run from SQLiteStudio



Test No	Test Description	Result	Test Data/Screenshots
2	<p>SQL query for primary key get.</p> <p>Check that sql for generation of table primary keys, formatting "AAA*-NNN" e.g. "USR-001", works to return the next value</p>	PASSED	below

Confirm pull of MAX value previously in table, then change to number to add value to otherwise possible error from string:

```
3 SELECT MAX(substr(part_id, 5)) FROM stock;
```

```
Grid view Form view
Total rows 1
MAX(substr(part id, 5))
1 016
```

```
3 SELECT MAX(CAST (substr(part_id, 5) AS INTEGER)) FROM stock;
```

```
Grid view Form view
Total rows loaded: 1
MAX(CAST (substr(part id, 5) AS INTEGER))
1 16
```

```
3 SELECT
4 printf('%03d', 1) as 'value'
5 ,part_id
6
7 FROM stock
8 WHERE part_id = 'ITM-001';
```

```
Grid view Form view
Total rows loaded: 1
value part id
1 001 ITM-001
```

```
1 SELECT CONCAT('AUD-',printf('%03d', COALESCE( (SELECT MAX(CAST (substr(audit_record_id, 5) AS INTEGER) ) FROM audit_table), 0 ) + 1));
2 SELECT 'AUD-' || printf('%03d', COALESCE( (SELECT MAX(CAST (substr(audit_record_id, 5) AS INTEGER) ) FROM audit_table), 0 ) + 1);
```

```
Grid view Form view
Total rows loaded: 1
CONCAT('AUD-',printf('%03d', COALESCE((SELECT MAX(CAST (substr(audit record id, 5) AS INTEGER) ) FROM au
1 AUD-165
```

Confirm display of full 3 digits for input of integer values under 100 & pull of the MAX of respective fieldname:

```
3 SELECT
4 printf('%03d', 1) as 'value'
5 ,MAX(part_id)
6
7 FROM stock;
```

```
Grid view Form view
Total rows loaded: 1
value MAX(part id)
1 001 ITM-016
```

CONCAT vs ||. Both return the same, || chosen as visually is cleaner for repeated function

Test No	Test Description	Result	Test Data/Screenshots
3	INSERT/UPDATE/DELETE in Single Table. Ensure that table(s) can be edited (following any constraint requirements - Happy Path)	PASSED	below

Insert & UPDATE script(s) into SQLiteStudio for user creation:

```

1 INSERT INTO users(
2     user_id, name, address, postcode_id, email, phone_no, primary_garage, access_code, account_creation_date, active_flag)
3 VALUES
4 ('USR-099', 'Amelia Jones', 'Flat 4, 22 King's Road, London', 'PST-001', 'amelia.jones@email.com', '07234 567890', 'N/A', 'CUS_USR', '2025-10-25', 1)
5 ;
6
7 UPDATE users SET
8 user_id = "USR-002", name = "Amelia Jones", address = "Flat 4, 22 King's Road, London",
9 postcode_id = "PST-002", email = "amelia.jones@email.com", phone_no = "07234 567890", primary_garage = "N/A", access_code = "CUS_USR"
, account_creation_date = "2025-10-25", active_flag = 1
10 WHERE user_id = "USR-002"
```

, constraint errors:

2	AUD-002	CREATE	USR-099	-
3	AUD-003	UPDATE	USR-002	{"name": "Amelia Jones", "address": "Flat 4, 22 King's Road, ...}
6	AUD-006	DELETE	USR-099	-
-	AUD-007	CREATE	USR-000	-

Query finished in 0.000 second(s).

Test No	Test Description	Result	Test Data/Screenshots
4	<p>INSERT into Db breaking Key constraints.</p> <p>Attempt INSERT error to confirm constraint functions as intended. UNQIUE / Non-existence postcode</p>	PASSED	below

Script to attempt input of postcode 999 which doesn't exist within the postcode table:

```

Structure Data Constraints
Grid view Form view
Query History
1 INSERT INTO users(
2     user_id, name, address, postcode_id, email, phone_no, primary_garage, access_code, account_creation_date, active_flag)
3 VALUES
4 ('USR-099', 'Amelia Jones', 'Flat 4, 22 King's Road, London', 'PST-999', 'amelia.jones@email.com', '07234 567890', 'N/A', 'CUS_USR', '2025-10-25', 1)
5 ;
6

Status
[17:30:57] Error while executing SQL query on database 'database0': FOREIGN KEY constraint failed
[17:31:04] Error while executing SQL query on database 'database0': FOREIGN KEY constraint failed

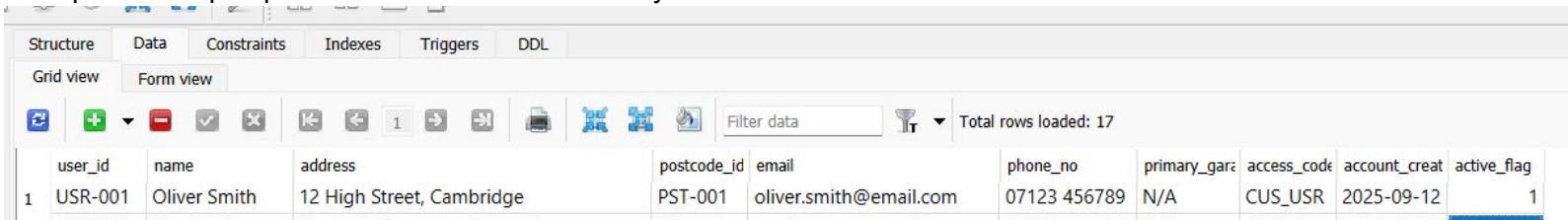
1 INSERT INTO users(
2     user_id, name, address, postcode_id, email, phone_no, primary_garage, access_code, account_creation_date, active_flag)
3 VALUES
4 ('USR-099', 'Amelia Jones', 'Flat 4, 22 King's Road, London', 'PST-001', 'amelia.jones@email.com', '07234 567890', 'N/A', 'RUBBISH', '2025-10-25', 1)
5 ;
6

Status
[17:35:48] Error while executing SQL query on database 'database0': FOREIGN KEY constraint failed
[17:38:49] Error while executing SQL query on database 'database0': FOREIGN KEY constraint failed

```

Test No	Test Description	Result	Test Data/Screenshots
5	INSERT into Db breaking UNQIUE constraints. Force INSERT error to confirm constraint functions as intended. Script attempting to create "USR-001" again	PASSED	below

Script to attempt input of "USR-001" which already exist within the user table:



	user_id	name	address	postcode_id	email	phone_no	primary_garage	access_code	account_creation_date	active_flag
1	USR-001	Oliver Smith	12 High Street, Cambridge	PST-001	oliver.smith@email.com	07123 456789	N/A	CUS-USR	2025-09-12	1

Sql return error as expected. UNIQUE constraint was broken no INSERT allowed:



```

Query History
1 INSERT INTO users(
2     user_id, name, address, postcode_id, email, phone_no, primary_garage, access_code, account_creation_date, active_flag)
3 VALUES
4 ('USR-001', 'Amelia Jones', 'Flat 4, 22 King's Road, London', 'PST-001', 'amelia.jones@email.com', '07234 567890', 'N/A', 'CUS-USR', '2025-10-25', 1)
5 ;
6
Status
[17:55:40] Error while executing SQL query on database 'database0': UNIQUE constraint failed: users.user_id
[17:55:41] Error while executing SQL query on database 'database0': UNIQUE constraint failed: users.user_id

```

Test No	Test Description	Result	Test Data/Screenshots
6	Trigger Creation & execution. Confirm creation without error	PASSED	below
7	Trigger INSERT/DELETE. Confirm trigger creation & required outputs without error (NEW only for INSERT, and OLD for DELETE)	PASSED	below

INSERT and Delete scripts run after trigger successful creation.

Returned success for both trigger creation & scripts, confirmation within audit table that update for changes (INSERT/DELETE) completed:

```
Query History
1 INSERT INTO users(
2     user_id, name, address, postcode_id, email, phone_no, primary_garage, access_code, account_creation_date, active_flag)
3 VALUES
4 ('USR-099', 'Amelia Jones', 'Flat 4, 22 King's Road, London', 'PST-001', 'amelia.jones@email.com', '07234 567890', 'N/A', 'CUS_USR', '2025-10-25
5 ;
6
7 DELETE FROM users WHERE user_id = 'USR-099';
```

The screenshot shows the Oracle SQL Developer interface. On the left, there is a code editor window containing the following PL/SQL code:

```
1 CREATE TRIGGER trg_users_after_delete
2     AFTER DELETE
3     ON users
4     FOR EACH ROW
5 BEGIN
6     INSERT INTO audit_table (
7         audit_record_id,
8         action_type,
9         primary_key_ref,
10        change_from,
11        change_to,
12        date_of_change,
13        time_of_change
14    )
15    VALUES (
16        'AUD-' || printf('%03d', COALESCE( (
17            SELECT MAX(CAST (substr(audit_record_id, 5) AS INTEGER)
18            FROM audit_table
19            ), 0) + 1),
20        'DELETE',
21        OLD.user_id,
22        NEW.user_id||'new' || OLD.user_id || 'name' || NEW.name || 'address' || OLD.address || 'postcode' || OLD.postcode_id
23    );
24 END;
25 /
```

To the right of the code editor is a results grid showing two rows of audit log entries:

Grid view	Form view
168 AUD-168 CREATE USR-099 -	Total rows loaded: 0
169 AUD-169 DELETE USR-099 {"user_id": "USR-099", "name": "Amelia Jones", "address": "Flat 4,	Status

168	AUD-168	CREATE	USR-099	-
169	AUD-169	DELETE	USR-099	{"user_id": "USR-099", "name": "Amelia Jones", "address": "Flat 4,

[17:39:53] Query finished in 0.012 second(s).

Test No	Test Description	Result	Test Data/Screenshots
8	<p>Trigger UPDATE.</p> <p>Confirm trigger creation & required outputs without error (NEW and OLD data pulled)</p>	PASSED	below

UPDATE scripts run removing “S” and then returning, after each trigger successfully executed into audit table for update:

```

1 INSERT INTO users(
2     user_id, name, address, postcode_id, email, phone_no, primary_garage, access_code, account_creation_date, active_flag)
3 VALUES
4 ('USR-002','Amelia Jone','Flat 4, 22 King's Road, London','PST-001','amelia.jones@email.com','07234 567890','N/A','CUS_USR','2025-10-25',1)
5 ;
6

```

166	AUD-166	UPDATE	USR-002	{"user_id": "USR-002", "name": "Amelia Jone", "address": "Flat 4, 22 King's Road, London", "postcode_id": "PST-002", "email": "amelia.jones@email.com", "phone_no": "07234 567890", "primary_garage": "N/A", "access_code": "CUS_USR", "account_creation_date": "2025-10-25", "active_flag": 1}	{"user_id": "USR-002", "name": "Amelia Jones", "address": "Flat 4, 22 King's Road, London", "postcode_id": "PST-002", "email": "amelia.jones@email.com", "phone_no": "07234 567890", "primary_garage": "N/A", "access_code": "CUS_USR", "account_creation_date": "2025-10-25", "active_flag": 1}	date_of_change	time_of_change
167	AUD-167	UPDATE	USR-002	{"user_id": "USR-002", "name": "Amelia Jones", "address": "Flat 4, 22 King's Road, London", "postcode_id": "PST-002", "email": "amelia.jones@email.com", "phone_no": "07234 567890", "primary_garage": "N/A", "access_code": "CUS_USR", "account_creation_date": "2025-10-25", "active_flag": 1}	{"user_id": "USR-002", "name": "Amelia Jones", "address": "Flat 4, 22 King's Road, London", "postcode_id": "PST-002", "email": "amelia.jones@email.com", "phone_no": "07234 567890", "primary_garage": "N/A", "access_code": "CUS_USR", "account_creation_date": "2025-10-25", "active_flag": 1}	2025-12-17	17:21:22
168	AUD-168	CREATE	USR-009			2025-12-17	17:25:06