DBMS PROJECT

PROJECT NAME: PlsLetMeGo

PROJECT TOPIC: Leave Management System

STUDENT NAME: Nakul Krishnakumar

ROLL NO: 2023BCS0010

GROUP NO: 3

Entities and Attributes

User

- User_id
- Password
- User_type

Student

- Student_ID
- Student_Name
- Email_ID
- Address
- Guardian_Phone_No

Faculty advisor

- FA_ID
- FA_Name
- Email_ID

Warden

- Warden_ID
- Warden_Name
- Email_ID

Gate

- Out_date
- In_date

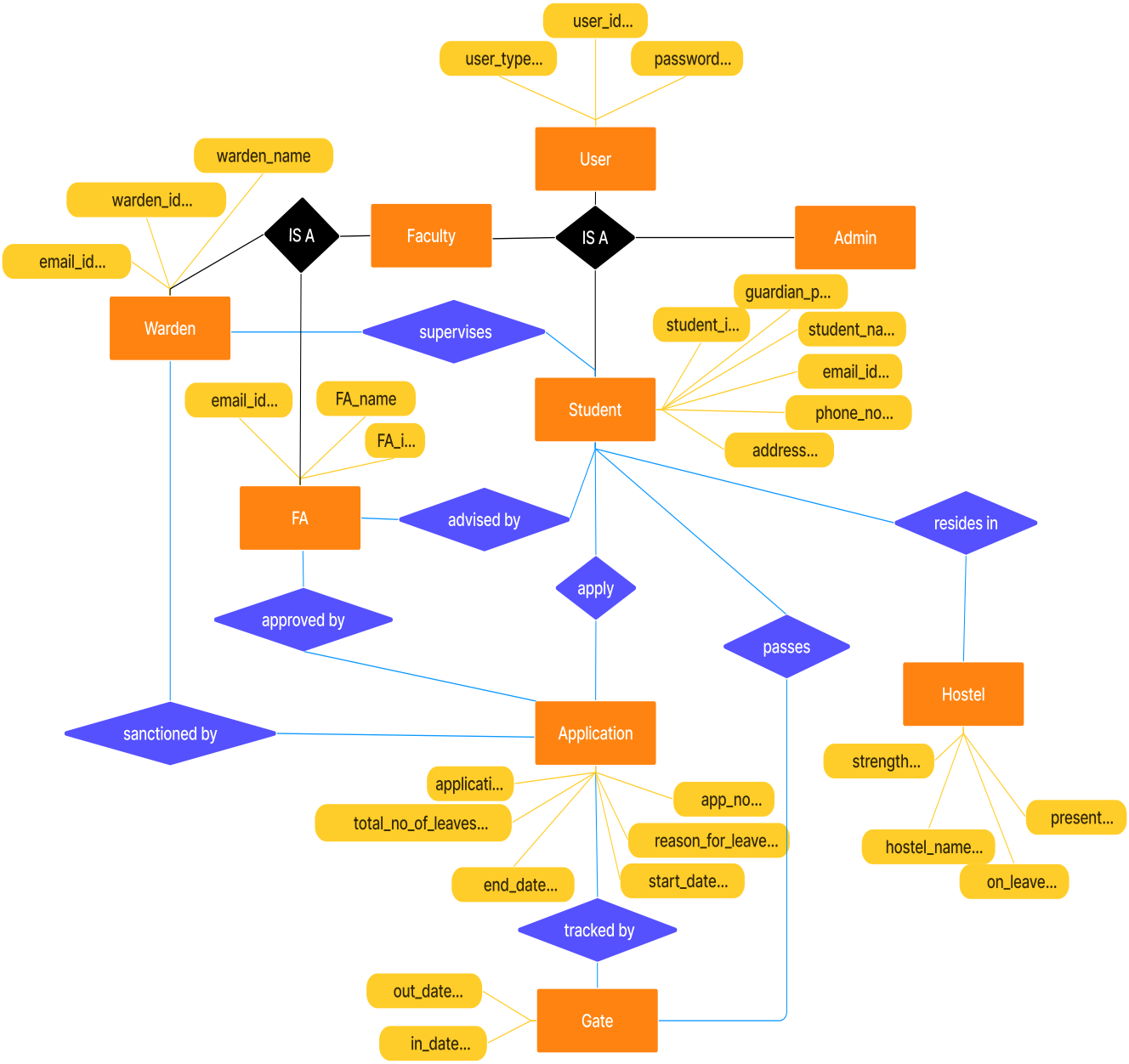
Application

- App_no
- Application_status
- Reason_for_leave
- Start_date
- End_date
- Total_no_of_working_days
-

Hostel

- Hostel_name
- Strength
- On_leave
- Present

ER Diagram


user_id...
user_type...
password...
User
warden_name
warden_id...
email_id...
IS A
Faculty
IS A
Admin
Warden
supervises
guardian_p...
student_i...
student_na...
email_id...
phone_no...
address...
Student
email_id...
FA_name
FA_i...
FA
advised by
resides in
apply
approved by
passes
Hostel
sanctioned by
Application
strength...
applicati...
app_no...
total_no_of_leaves...
reason_for_leave...
present...
hostel_name...
end_date...
start_date...
on_leave...
tracked by
out_date...
Gate
in_date...

Relations and their Cardinality and Participation

STUDENT apply APPLICATION

- **Cardinality (One-to-many):** Each student can submit multiple applications, but each application is linked to only one student.
- **Participation (Partial-partial):** Not every student submits an application, and not all applications are submitted.

STUDENT advised by FA

- **Cardinality (Many-to-one):** Multiple students may share the same faculty advisor, but each student has only one assigned advisor.
- **Participation (Total-total):** Every student must have an advisor, and each advisor is responsible for students.

APPLICATIONS approved by FA

- **Cardinality (Many-to-one):** Each application is approved by only one faculty advisor, but an advisor can approve multiple applications.
- **Participation (Total-partial):** All applications require approval, but not every advisor needs to approve applications.

APPLICATIONS sanctioned by WARDEN

- **Cardinality (Many-to-many):** Multiple wardens can sanction the same application, and a warden can sanction multiple applications.
- **Participation (Total-partial):** All applications are eligible for sanction, but not every application will require it.

APPLICATIONS tracked by GATE

- **Cardinality (One-to-one):** Each application is uniquely tracked by a specific gate, and each gate entry tracks a specific application.
- **Participation (Total-total):** Every application has a corresponding gate entry, and all gate entries are linked to applications.

STUDENT passes GATE

- **Cardinality (One-to-many):** Each student can pass through the gate multiple times, but each gate entry is linked to only one student.
- **Participation (Total-partial):** All students have the opportunity to pass through, though some may not.

STUDENT resides in HOSTEL

- **Cardinality (Many-to-one):** Multiple students may stay in the same hostel, but each student resides in only one hostel at a time.
- **Participation (Partial-partial):** Not all students are required to stay in a hostel, and not all hostels may be occupied by students.

USER represent STUDENT/FACULTY/ADMIN

- **Cardinality (One-to-one):** Each user account uniquely represents one entity (student, faculty, or admin), with no overlap.
- **Participation (Total-total):** Every user must represent exactly one role, and each role requires a corresponding user.

WARDEN supervises STUDENT

- **Cardinality (Many-to-many):** Each warden can supervise multiple students, and each student can be supervised by multiple wardens.
- **Participation (Total-total):** Every student must have a supervising warden, and each warden must oversee students in the hostel system.

Relational Schema

USER (User_id, Password, User_type)

STUDENT (Student_id, Student_name, Email_ID, Phone_No, Guardian_Phone_No, FA_id)

FA(FA_id, FA_Name, Email_ID)

WARDEN(Warden_id, Warden _Name, Email_ID, Hostel_Name)

APPLICATION(app_no, reason, start_date, end_date, no_working_days, application_status, student_id)

GateEntry(entry_id, app_no, out_date, in_date)

Hostel(hostel_name strength on_leave)

Application_Warden(app_no warden_id)

Student_Warden(student_id warden_id)

Functional Dependencies

USER	user_id → password user_id → user_type
STUDENT	student_id → (student_name, email_id, phone_no, address, guardian_phone_no, FA_id, hostel_name) email_id → (student_id, student_name, phone_no, address, guardian_phone_no, FA_id, hostel_name) phone_no → (student_id, student_name, address, email_id, guardian_phone_no, FA_id, hostel_name)
FACULTY_ADVISOR	fa_id →(fa_name, email_id) email_id → (fa_id, fa_name)
WARDEN	warden_id → (wd_name, email_id, hostel_name)
APPLICATION	app_no → (reason, start_date, end_date, no_working_days, application_status, student_id, fa_id) student_id → fa_id

GATE_ENTRY	entry_id → (app_no, out_date, in_date) app_no → (out_date, in_date)
HOSTEL	hostel_name → (strength, on_leave)
APPLICATION_WARDEN	(app_no, warden_id) → (app_no, warden_id)
STUDENT_WARDEN	(student_id, warden_id) → (student_id, warden_id)

NORMALIZATION APPLIED

USER :

- Table is already in 1NF as no multivalued attribute exists.
- Both user_type and password are fully dependent on candidate key (user_id), hence the table is in 2NF.
- In all functional dependency, the LHS is candidate (which is subset of super key), hence table is in 3NF.

STUDENT :

- Table is already in 1NF as no multivalued attribute exists.
- Since each candidate key is a single attribute, there is no possibility of partial dependency. Each non-prime attribute depends on a whole candidate key.
- In each case, non-prime attributes (student_name, address, guardian_phone, FA_id, hostel_name) are directly dependent on the candidate keys and not on any other non-prime attribute.Thus, there are no transitive dependencies, and the table is in 3NF.

FACULTY_ADVISOR:

- Table is already in 1NF as no multivalued attribute exists.
- Since both candidate keys (fa_id and email_id) are single attributes, there can't be any partial dependency (as partial dependency only occurs with composite keys). fa_name is fully functionally dependent on each candidate key (fa_id and email_id), so there are no partial dependencies. Conclusion: The table is in 2NF.
- Here, we have two FDs: fa_id → fa_name, email_id and email_id → fa_id, fa_name. Both fa_id and email_id are superkeys (candidate keys) for this table. The non-prime attribute fa_name is directly dependent on candidate keys without any intermediary dependencies. Conclusion: The table is in 3NF.
- Both FDs (fa_id → fa_name, email_id and email_id → fa_id, fa_name) have superkeys on the left side (fa_id and email_id are candidate keys), which satisfies BCNF.

WARDEN:

- Table is already in 1NF as no multivalued attribute exists.
- Since there is only one candidate key (warden_id), and all non-prime attributes are fully dependent on warden_id, this table is in 2NF.
- In this case, there are no transitive dependencies. All non-prime attributes depend directly on the candidate key (warden_id), so the table is in 3NF.
- Here, the only functional dependency is warden_id → (wd_name, email_id, hostel_name). Since warden_id is a candidate key (and hence a super key), the table is also in BCNF.

APPLICATION :

- Table is already in 1NF as no multivalued attribute exists.
- Here, fa_id is only dependent on student_id and not directly on app_no, creating a partial dependency. To bring the table into 2NF, we need to remove this partial dependency by separating fa_id into a different table. Since student table contains fa_id we remove it from the application table
- The table is in 3NF form as there are no transitive dependency present.

GATE_ENTRY:

- Table is already in 1NF as no multivalued attribute exists.
- In this case, all non-prime attributes (app_no, out_date, in_date) are fully dependent on entry_id, the candidate key.Therefore, the GateEntry table is already in 2NF.

HOSTEL:

- Table is already in 1NF as no multivalued attribute exists.
- Here, strength and on_leave are fully dependent on hostel_name, which is the only candidate key. Therefore, the Hostel table is already in 2NF.
- In this table, there are no transitive dependencies. strength and on_leave are directly dependent on hostel_name, the candidate key. Therefore, the Hostel table is already in 3NF.
- Since hostel_name is the only candidate key and all functional dependencies have hostel_name as their determinant, the table is also in BCNF.

APPLICATION_WARDEN:

- Table is already in 1NF as no multivalued attribute exists.
- All attributes (in this case, app_no and warden_id as the composite key) are fully functionally dependent on the entire candidate key.
- There are no transitive dependencies, as there are no non-key attributes.
- The table is also in BCNF because: For every functional dependency, the determinant is a superkey. Here, the only functional dependency is the primary key itself, which is also a superkey.

STUDENT_WARDEN:

- Table is already in 1NF as no multivalued attribute exists.
- All attributes (in this case, student_id and warden_id as the composite key) are fully functionally dependent on the entire candidate key.
- There are no transitive dependencies, as there are no non-key attributes.
- The table is also in BCNF because: For every functional dependency, the determinant is a superkey. Here, the only functional dependency is the primary key itself, which is also a superkey.

VALUES ENTERED INTO TABLES

TABLE -Users

user_id varchar	username varchar	password varchar	user_type varchar
2023BCD0058	Adwaith E S	$2a$10$BfY.hCeuikCZ0nyVslzf/.WbctS2F	student
2023BCS0010	Nakul Krishnakumar	$2a$10$zHIeiFKAi1Ij8lVD1.T0mOplA4oha	student
2023BCS0142	Vigil Das	$2a$10$Sa.ZudT98qx8wajFSL9XnOksZ0I	student
2023BCS0184	Rithika E	$2a$10$WTEOmYpeq6jIjaUSqVTK0OdyF	student
2023BCS0187	Aadhil Rahman	$2a$10$izcPXCOEgkvm3GEEskanJOZ43	student
2023BCS0190	Leah Mary Mathew	$2a$10$LkpBzlE1rYs/xAwPX74azeTd97rxt	student
2023BCY0040	Vaisag J	$2a$10$sp9/utWqSd635i4ViodiHukpPLJ	student
2023BCY0041	Manasa Manoj	$2a$10$qyx/s5DaFlmcnCBa4iZ3M.mfnvy	student
admin	admin	$2a$10$TQkUX3a/aRBEzDT.x5zgh.mU82	admin
FAC0001	Dr. Anjali Menon	$2a$10$y7f2RZq8fmf8H/M/c.EUH.BLNpt	fa
FAC0002	Dr. Rajesh Iyer	$2a$10$sicUxYA3avxyJ7fjaGRNBuVZgIF	fa
FAC0003	Dr. Priya Varma	$2a$10$4SYOUFITHJCUFaoowtVjDeJQB	fa
FAC0004	Dr. Sunil Kumar	$2a$10$9wSPIhHUpk3FFS112e7CvexhGZ	fa
FAC0005	Dr. Meera Sharma	$2a$10$9UPXRdIefkMT.AsejuiYiO4oBGy	fa
FAC0006	Dr. Ramesh Gupta	$2a$10$1SwPAtWNxn7bL.I.7GoAyeIL0Fv	fa
FAC0007	Dr. Anitha Patel	$2a$10$ENf2mPGgpDYv6HarxjIJBu1bKv	fa
gate	gate	$2a$10$.FeXDmP1EQxYVRgJqMU.Rum/d	gate
WRD0001	Mr. John Mathew	$2a$10$2MikAOhsdsFH6dMJb/3TxO6Ep	warden
WRD0002	Ms. Neeta Roy	$2a$10$YWmqVKB0gJxqLSAcbBUgw.kr	warden
WRD0003	Mr. Ravi Shankar	$2a$10$57Bamyzo5Wk1YxAAnxM3cORT	warden
WRD0004	Mrs. Latha Suresh	$2a$10$iE61JbwgGACYwJNbW/pJ8uPF	warden
WRD0005	Ms. Kamala Devi	$2a$10$mxOVBfj/PppxhiV8MkC5oelOdv	warden
WRD0006	Mr. Mohan Pillai	$2a$10$BVE4PfhzOlufgKLYDq1c8ugMkF	warden
WRD0007	Mrs. Sheila Thomas	$2a$10$e55N7qZEM/HkZJnvTS189.FHjU	warden

TABLE-FacultyAdvisor

fa_id varchar	fa_name varchar	email_id varchar
FAC0001	Dr. Anjali Menon	anjalimenon@gmail.com
FAC0002	Dr. Rajesh Iyer	rajeshiyer@gmail.com
FAC0003	Dr. Priya Varma	priyavarma@gmail.com
FAC0004	Dr. Sunil Kumar	sunilkumar@gmail.com
FAC0005	Dr. Meera Sharma	meerasharma@gmail.com
FAC0006	Dr. Ramesh Gupta	rameshgupta@gmail.com
FAC0007	Dr. Anitha Patel	anithapatel@gmail.com

TABLE-Warden

warden_id varchar	wd_name varchar	email_id varchar	hostel_name varchar
WRD0001	Mr. John Mathew	johnmathew@gmail.com	Agastya
WRD0002	Ms. Neeta Roy	neetaroy@gmail.com	Cooptyre
WRD0003	Mr. Ravi Shankar	ravishankar@gmail.com	Manimala
WRD0004	Mrs. Latha Suresh	lathasuresh@gmail.com	Anamudi
WRD0005	Ms. Kamala Devi	kamaladevi@gmail.com	Meenachil
WRD0006	Mr. Mohan Pillai	mohanpillai@gmail.com	Nila
WRD0007	Mrs. Sheila Thomas	sheilathomas@gmail.com	Sahayadri

TABLE – GateEntry

entr... i...	out_d... timesta...	in_date timestamp	app... i...
15	2024-11-03 14:44:00	2024-11-03 14:51:00	1
16	2024-11-03 14:51:00	2024-11-03 14:51:00	3
17	2024-11-03 14:53:00	NULL	4
18	2024-11-03 15:06:00	2024-11-03 15:07:00	5

TABLE-Hostel

hostel_name varchar	strength int4	on_leave int4
Agastya	100	1
Anamudi	150	0
Cooptyre	90	0
Manimala	110	0
Meenachil	120	0
Nila	70	0
Sahayadri	80	0

Table-Student

student_id varchar	email_id varchar	phone_no int8	address varchar	guardian_phone int8	fa_id varchar	hostel_name varchar
2023BCD0058	adwaithes@gmail.com	5551234567	456 Oak Ave, Metropolis, NY 10001	4443210987	FAC0002	Sahayadri
2023BCS0010	nakulkrishnakumar@gmail.com	9876543210	123 Main St, Springfield, IL 62701	5559876543	FAC0001	Agastya
2023BCS0184	rithikae@gmail.com	1234567890	789 Pine Rd, Gotham, NJ 07001	8887654321	FAC0003	Anamudi
2023BCS0187	aadhilrahman@gmail.com	2345678901	654 Maple Blvd, Central City, TX 75001	7775551212	FAC0005	Sahayadri
2023BCS0190	leahmarymathew@gmail.com	8765432109	321 Elm St, Star City, CA 90001	2223334444	FAC0004	Anamudi
2023BCY0040	vaisagj@gmail.com	3456789012	987 Cedar Lane, Smallville, KS 67501	9998887777	FAC0006	Sahayadri
2023BCY0041	manasamanoj@gmail.com	4123456789	135 Cherry St, Riverdale, GA 30274	3334445555	FAC0007	Anamudi

Table – Application

app... i...	student_id varchar	start_date timestamp	end_d... timesta...	reason text	leave_addr text	no_of_working_days int4	app_status status
1	2023BCS0010	2024-11-07 17:00:00	2024-11-11 07:00:00	Home Visit	"KRISHNA", Behind KAP Camp, Punnakul	1	Expired
2	2023BCS0010	2024-11-12 17:00:00	2024-11-17 20:00:00	Alan Walker Concert	Fort Kochi, Kochi, Ernakulam	3	Rejected
3	2023BCS0010	2024-11-03 14:43:00	2024-11-04 14:43:00	Internship	Kottayam City, Kottayam, Kerala	0	Expired
4	2023BCS0010	2024-11-03 14:52:00	2024-11-04 14:52:00	Home Visit	KRISHNA, Behind KAP Camp, Kannur	0	On Leave
5	2023BCS0184	2024-11-08 13:02:00	2024-11-11 07:02:00	Home visit	Erayil House, Karikkankulam, PO Pappini	0	Expired
6	2023BCS0184	2024-11-16 06:30:00	2024-11-17 17:00:00	Visiting Kochi	Erayil House, Karikkankulam, Pappinisser	0	Rejected
7	2023BCS0184	2024-11-03 17:00:00	2024-11-06 07:00:00	Medical Reason, Home Visit	Erayil House, Karikkankulam, Pappinisser	2	Approved
8	2023BCS0184	2024-11-03 17:00:00	2024-11-06 07:00:00	Medical Reason, Home Visit	Erayil House, Karikkankulam, Pappinisser	2	Approved

Table – Application_warden

app_no int4	warden_id varchar
1	WRD0001
2	WRD0001
3	WRD0001
4	WRD0001
5	WRD0004
6	WRD0004
7	WRD0004
8	WRD0004

Table – Student_warden

student_id varchar	warden_id varchar
2023BCD0058	WRD0007
2023BCS0010	WRD0001
2023BCS0184	WRD0004
2023BCS0187	WRD0007
2023BCS0190	WRD0004
2023BCY0040	WRD0007
2023BCY0041	WRD0004

TRIGGERS USED

TRIGGER 1	Trigger to Reset Application Number to last tuple's app_no – 1 or set it to 0 if application table is empty
	
	CREATE OR REPLACE FUNCTION reset_app_no()
	RETURNS TRIGGER AS $$
	DECLARE
	  max_app_no INT;
	BEGIN
	  -- Check if the table is empty
	  SELECT COUNT(*) INTO max_app_no FROM application;
	
	  IF max_app_no = 0 THEN
	  -- If empty, set app_no to 1 for the new row
	  NEW.app_no := 1;
	  ELSE
	  -- Otherwise, find the next available app_no
	  SELECT COALESCE(MAX(app_no), 0) + 1 INTO NEW.app_no FROM application;
	  END IF;
	
	  RETURN NEW; -- Return the modified new row with updated app_no
	END;
	$$ LANGUAGE plpgsql;
	
	CREATE TRIGGER reset_app_no_trigger
	BEFORE INSERT ON application
	FOR EACH ROW
	EXECUTE FUNCTION reset_app_no();

TRIGGER 2	Trigger to add entries into application_warden table when a new application is made CREATE OR REPLACE FUNCTION add_application_warden() RETURNS TRIGGER AS $$ BEGIN

	INSERT INTO application_warden (app_no, warden_id) SELECT NEW.app_no, warden.warden_id FROM warden JOIN student_warden ON student_warden.warden_id = warden.warden_id WHERE student_warden.student_id = NEW.student_id; RETURN NEW; END; $$ LANGUAGE plpgsql; CREATE TRIGGER trigger_add_application_warden AFTER INSERT ON application FOR EACH ROW EXECUTE FUNCTION add_application_warden();

TRIGGER 3	:Increment On_leave attribute in hostel table when an application is checked out of gate
	
	CREATE OR REPLACE FUNCTION increment_on_leave()
	RETURNS TRIGGER AS $$
	BEGIN
	-- Check if the application status has changed to "On Leave"
	IF NEW.app_status = 'On Leave' AND OLD.app_status IS DISTINCT FROM 'On Leave' THEN
	UPDATE Hostel
	SET on_leave = on_leave + 1
	WHERE hostel_name = (
	SELECT hostel_name FROM Student WHERE student_id = NEW.student_id
	);
	END IF;
	
	RETURN NEW;
	END;
	$$ LANGUAGE plpgsql;
	
	CREATE TRIGGER trigger_increment_on_leave
	AFTER UPDATE OF app_status ON Application
	FOR EACH ROW
	WHEN (NEW.app_status = 'On Leave')
	EXECUTE FUNCTION increment_on_leave();
	
	DROP TRIGGER trigger_increment_on_leave ON application;

TRIGGER 4	Decrement On_leave attribute when an application is checked in at the gate CREATE OR REPLACE FUNCTION decrement_on_leave() RETURNS TRIGGER AS $$ BEGIN -- Check if the application status has changed to "Late" or "Expired"

	IF (NEW.app_status = 'Late' OR NEW.app_status = 'Expired')
	AND OLD.app_status IS DISTINCT FROM NEW.app_status THEN
	UPDATE Hostel
	SET on_leave = on_leave - 1
	WHERE hostel_name = (
	SELECT hostel_name FROM Student WHERE student_id = NEW.student_id
	);
	END IF;
	
	RETURN NEW;
	END;
	$$ LANGUAGE plpgsql;
	
	CREATE TRIGGER trigger_decrement_on_leave
	AFTER UPDATE OF app_status ON Application
	FOR EACH ROW
	WHEN (NEW.app_status = 'Late' OR NEW.app_status = 'Expired')
	EXECUTE FUNCTION decrement_on_leave();

TRIGGER 5	:Add entry into student_warden table when a new student is registered (according to their hostel_name)
	
	CREATE OR REPLACE FUNCTION add_student_warden_entry()
	RETURNS TRIGGER AS $$
	DECLARE
	wardenID varchar(50);
	BEGIN
	-- Retrieve the warden_id associated with the new student's hostel
	SELECT warden_id INTO wardenID
	FROM Warden
	WHERE hostel_name = NEW.hostel_name;
	
	-- Insert into student_warden if a matching warden_id is found
	IF wardenID IS NOT NULL THEN
	INSERT INTO student_warden (student_id, warden_id)
	VALUES (NEW.student_id, wardenID);
	END IF;
	
	RETURN NEW;
	END;
	$$ LANGUAGE plpgsql;
	
	CREATE TRIGGER after_student_insert
	AFTER INSERT ON Student
	FOR EACH ROW
	EXECUTE FUNCTION add_student_warden_entry();

QUESTIONS

-- 1. Find all applications that were approved for medical reasons

```
SELECT a.app_no, a.student_id, a.reason, a.start_date, a.end_date
FROM APPLICATION a
WHERE a.reason LIKE '%Medical%'
AND a.app_status = 'Approved';
```

-- 2. Count the number of applications handled by each warden

```
SELECT w.warden_id, w.wd_name, COUNT(aw.app_no) as applications_handled
FROM WARDEN w
LEFT JOIN Application_Warden aw ON w.warden_id = aw.warden_id
GROUP BY w.warden_id, w.wd_name;
```

-- 3. List students who have more than 2 rejected applications

```
SELECT s.student_id, s.email_id, COUNT(*) as rejected_count
FROM STUDENT s
JOIN APPLICATION a ON s.student_id = a.student_id
WHERE a.app_status = 'Rejected'
GROUP BY s.student_id, s.email_id
HAVING COUNT(*) > 2;
```

-- 4. Find the warden who has approved the most leave applications

```
SELECT w.warden_id, w.wd_name, COUNT(*) as approved_count
FROM WARDEN w
JOIN Application_Warden aw ON w.warden_id = aw.warden_id
JOIN APPLICATION a ON aw.app_no = a.app_no
WHERE a.app_status = 'Approved'
```

```
GROUP BY w.warden_id, w.wd_name
ORDER BY approved_count DESC
LIMIT 1;
```

-- 5. Calculate the average duration of approved leaves for each hostel

```
SELECT h.hostel_name,
    AVG(DATEDIFF(a.end_date, a.start_date)) as avg_leave_duration
FROM Hostel h
JOIN STUDENT s ON h.hostel_name = s.hostel_name
JOIN APPLICATION a ON s.student_id = a.student_id
WHERE a.app_status = 'Approved'
GROUP BY h.hostel_name;
```

-- 6. List all hostels that have strength greater than 100

```
SELECT hostel_name, strength
FROM Hostel
WHERE strength > 100;
```

-- 7. List wardens and their students who have pending applications

```
SELECT w.warden_id, w.wd_name, s.student_id, a.app_no, a.reason
FROM WARDEN w
JOIN Student_Warden sw ON w.warden_id = sw.warden_id
JOIN STUDENT s ON sw.student_id = s.student_id
JOIN APPLICATION a ON s.student_id = a.student_id
WHERE a.app_status = 'Pending';
```

-- 8. Display the name and email of warden who manages Nila hostel

```
SELECT wd_name, email_id
FROM WARDEN
WHERE hostel_name = 'Nila';
```

-- 9. Insert a new warden record

INSERT INTO WARDEN (warden_id, wd_name, email_id, hostel_name)

VALUES ('WRD0008', 'Mr. James Wilson', 'jameswilson@gmail.com', 'Nila');

-- 10. Update the email address of a student

UPDATE STUDENT

SET email_id = 'newemail@gmail.com'

WHERE student_id = '2023BCS0010';

-- 11. Update application status to 'Expired' for all past leaves

UPDATE APPLICATION

SET app_status = 'Expired'

WHERE end_date < CURDATE()

AND app_status = 'Approved';

-- 12. Find hostels with no students on leave

SELECT hostel_name

FROM Hostel

WHERE on_leave = 0;

-- 13. Find students who have never applied for leave but are assigned to a warden

SELECT s.student_id, s.email_id, w.warden_id, w.wd_name

FROM STUDENT s

JOIN Student_Warden sw ON s.student_id = sw.student_id

JOIN WARDEN w ON sw.warden_id = w.warden_id

LEFT JOIN APPLICATION a ON s.student_id = a.student_id

WHERE a.app_no IS NULL;

-- 14. List applications that were approved on the same day they were submitted

SELECT app_no, student_id, start_date, reason

```
FROM APPLICATION
WHERE DATE(start_date) = DATE(end_date)
AND app_status = 'Approved';

-- 15. Transfer all students from one warden to another
BEGIN TRANSACTION;
UPDATE Student_Warden
SET warden_id = 'WRD0002'
WHERE warden_id = 'WRD0001';

UPDATE Application_Warden
SET warden_id = 'WRD0002'
WHERE warden_id = 'WRD0001'
AND app_no IN (
    SELECT app_no
    FROM APPLICATION
    WHERE app_status = 'Pending'
);
COMMIT;
```