

Lab - 9 DB_Project_Assignment_ERD

Normalization and Schema Refinement,

DDL Codes and SQL Queries

Name: Ronit Jain (202001081)

Nipun Shah (202001096)

Group No: 2

Section No: 6

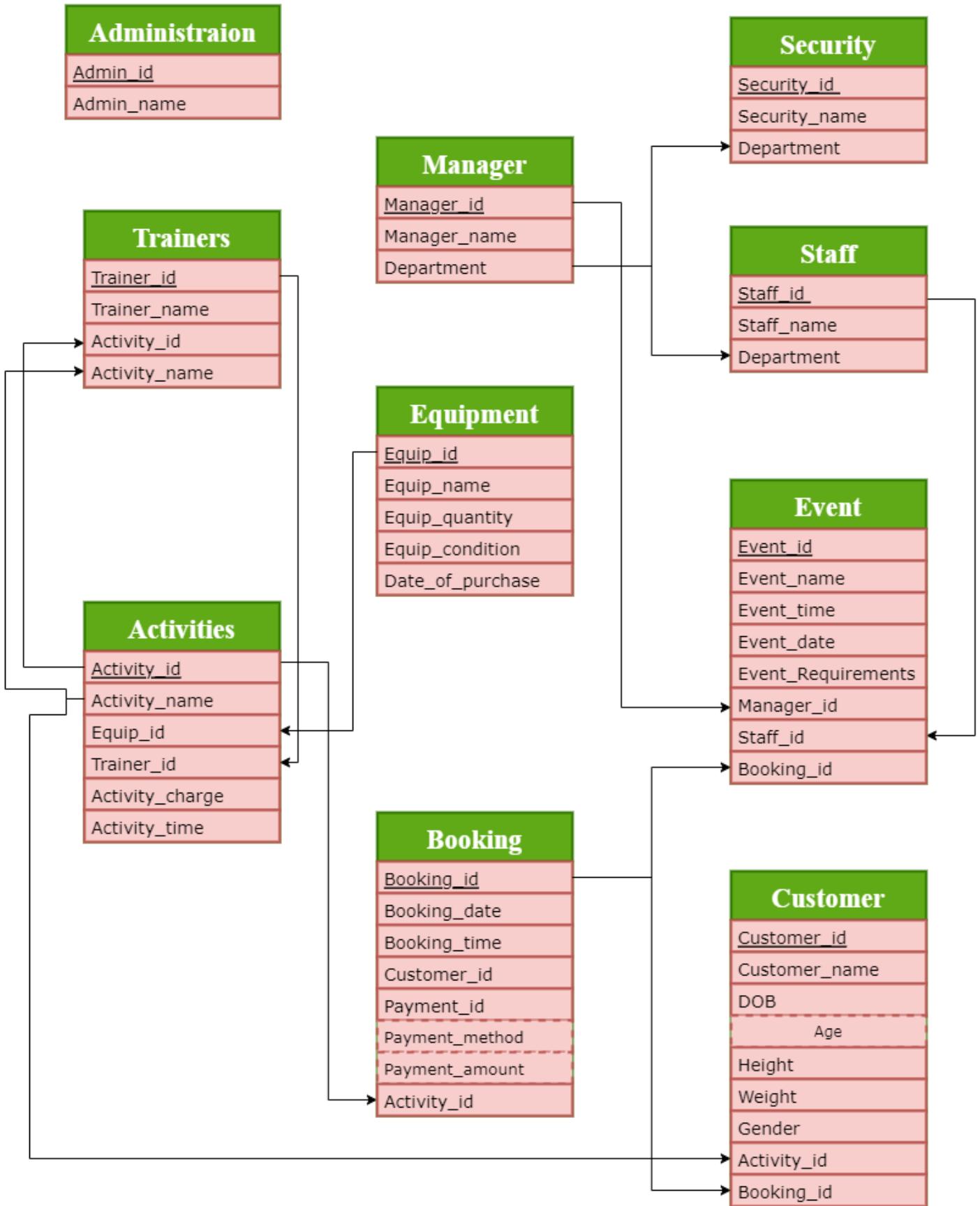
Team Id: 6.1

Case Study

Beach Activity Management System:

All beach activity-related records should be created in this database. The beach-related activities like scuba diving, snorkeling, paragliding, flyboarding, windsurfing, and so on. Precautionary things like helmets, swimsuits, quality shoes, gloves, etc., for particular activities, must be appropriately managed for all customers. The other things related to taking pictures and videos are also included in this database.

Normalization & Schema Refinement:



1. Administration (Admin_id, Admin_name)

Primary Key: Admin_id

Dependencies: Admin_id \rightarrow Admin_name

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: In Admin_name, it is possible that the same name is associated with a different Admin_id.

Anomalies:

Insert – None.

Update – None.

Delete – None.

Every attribute in this schema is single-valued (scalar), already in 1NF. There is no partial dependency here, so it is in 2NF.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in **BCNF**,

- a. It should be in the third normal form (3NF).
- b. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well.

2. Manager (Manager_id, Manager_name, department)

Primary Key: Manager_id

Dependencies:

Manager_id \rightarrow Manager_name,

Manager_id \rightarrow Department

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: In Manager_name, it is possible that the same name is associated with a different Manager_id. It might be possible that two manager_id are associated with the same department.

Anomalies:

Insert – None.

Update – None.

Delete – None.

In this schema, every attribute is single-valued (scalar), making it already in 1NF. There is no partial dependency here, so it is in 2NF.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in **BCNF**,

- a. It should be in the third normal form (3NF).
- b. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well.

3. Customer (Customer_id, Customer_name, DOB, age, Height, Weight, Gender, Activity_id, Booking_id)

Primary Key: Customer_id

Foreign Key: Activity_id, Booking_id

Dependencies:

Customer_id → Customer_name,

Customer_id → DOB,

Customer_id → Age,

DOB → Age,

Customer_id → Height,

Customer_id → Weight,

Customer_id → Gender,

Partial Key Dependency: None

Transitive Dependency: Age is derived from DOB, and DOB is derived from Customer_id. We will drop the age attribute as it can be derived from the date of birth (DOB).

Redundancies: In Customer_name, it is possible that the same name is associated with a different Customer_id.

Anomalies:

Insert – None.

Update – A customer's DOB cannot be updated once set.

Delete – Once assigned, one cannot delete the DOB or Activity_id of a particular customer.

In this, we drop booking_id, as in the booking table there is customer_id through which we can get all the booking details.

In this schema, every attribute is single-valued (scalar), making it already in 1NF. There is no partial dependency here, so it is in 2NF.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in **BCNF**,

- a. It should be in the third standard form (3NF).
- b. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well.

4. Booking (Booking_id, Booking_date, Booking_time, Customer_id, Payment_id, Payment_method, Payment_amount, Activity_id)

Primary Key: Booking_id

Foreign Key: Customer_id, Activity_id

Dependencies:

Booking_id → Booking_date,
Booking_id → Booking_time,
Booking_id → Payment_id,
Booking_id → Payment_method,
Booking_id → Payment_amount

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: None

Anomalies:

Insert – None.
Update – Once assigned booking_id cannot be updated.
Delete – None.

In this schema, every attribute is single-valued (scalar) making it already in 1NF. There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in BCNF,

- a. It should be in the third normal form (3NF).
- b. For any dependency A → B, A must be a super key.

Hence, this relation is in BCNF as well.

5. Activity (Activity_id, Activity_name, Activity_charge, Activity_time, Equip_id, Trainer_id)

Primary Key: Activity_id

Foreign Key: Equip_id, Trainer_id

Dependencies:

Activity_id \rightarrow Activity_name,

Activity_id \rightarrow Activity_charge

Activity_id \rightarrow Activity_time

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: None

Anomalies:

Insert – None.

Update – None

Delete – None.

In this schema, every attribute is not a single-valued (scalar) making it already in 1NF. There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in BCNF,

- a. It should be in the third normal form (3NF).
- b. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well.

6. Equipment (Equip_id, Equip_name, Equip_quantity, Equip_condition, Date_of_purchase)

Primary Key: Equip_id

Dependencies:

Equip_id-> Equip_name,
Equip_id-> Equip_quantity,
Equip_id-> Equip_condition
Equip_id -> Date_of_purchase

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: None

Anomalies:

Insert – None.

Update – None.

Delete – None.

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in **BCNF**,

- a. It should be in the third normal form (3NF).
- b. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well

7. Trainers (Trainer_id, Trainer_name, Activity_id, activity_name)

Primary Key: Trainer_id

Foreign Key: Activity_id

Dependencies:

Trainer_id-> Trainer_name

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: None

Anomalies:

Insert – None.

Update – None.

Delete – None.

In this, we drop activity_id and activity_name as this column is present in the Activity table and trainer_id is given to them as a foreign key.

In this schema, every attribute is single-valued (scalar) making it already in 1NF. There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in **BCNF**,

- a. It should be in the third normal form (3NF).
- b. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well

8. Event (Event_id, Event_name, Event_time, Event_date, Event_requirements, manager_id, Staff_id, Booking_id)

Primary Key: Event_id

Foreign Key: Manager_id, Booking_id, Staff_id

Dependencies:

Event_id \rightarrow Event_name,
Event_id \rightarrow Event_time,
Event_id \rightarrow Event_date,
Event_id \rightarrow Event_requirements

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: None

Anomalies:

Insert – None.

Update – None.

Delete – None.

In this schema, every attribute is single-valued (scalar) making it already in 1NF. There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in BCNF,

- a. It should be in the third normal form (3NF).
- b. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well

9. Security (Security_id, Security_name, Department)

Primary Key: Security_id

Dependencies:

Security_id → Security_name,
Security_id → Department

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: None

Anomalies:

Insert – None.

Update – None.

Delete – None.

In this schema, every attribute is single-valued (scalar) making it already in 1NF. There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in **BCNF**,

1. It should be in the third normal form (3NF).
2. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well

10. Staff (Staff_id, Staff_name, department)

Primary Key: Staff_id

Dependencies:

Staff_id \rightarrow Staff_name,
Staff_id \rightarrow Department

Partial Key Dependency: None

Transitive Dependency: None

Redundancies: None

Anomalies:

Insert – None.

Update – None.

Delete – None.

In this schema, every attribute is single-valued (scalar), making it already in 1NF. There is no partial dependency here, so it is in 2NF.

2NF Redundancies: None

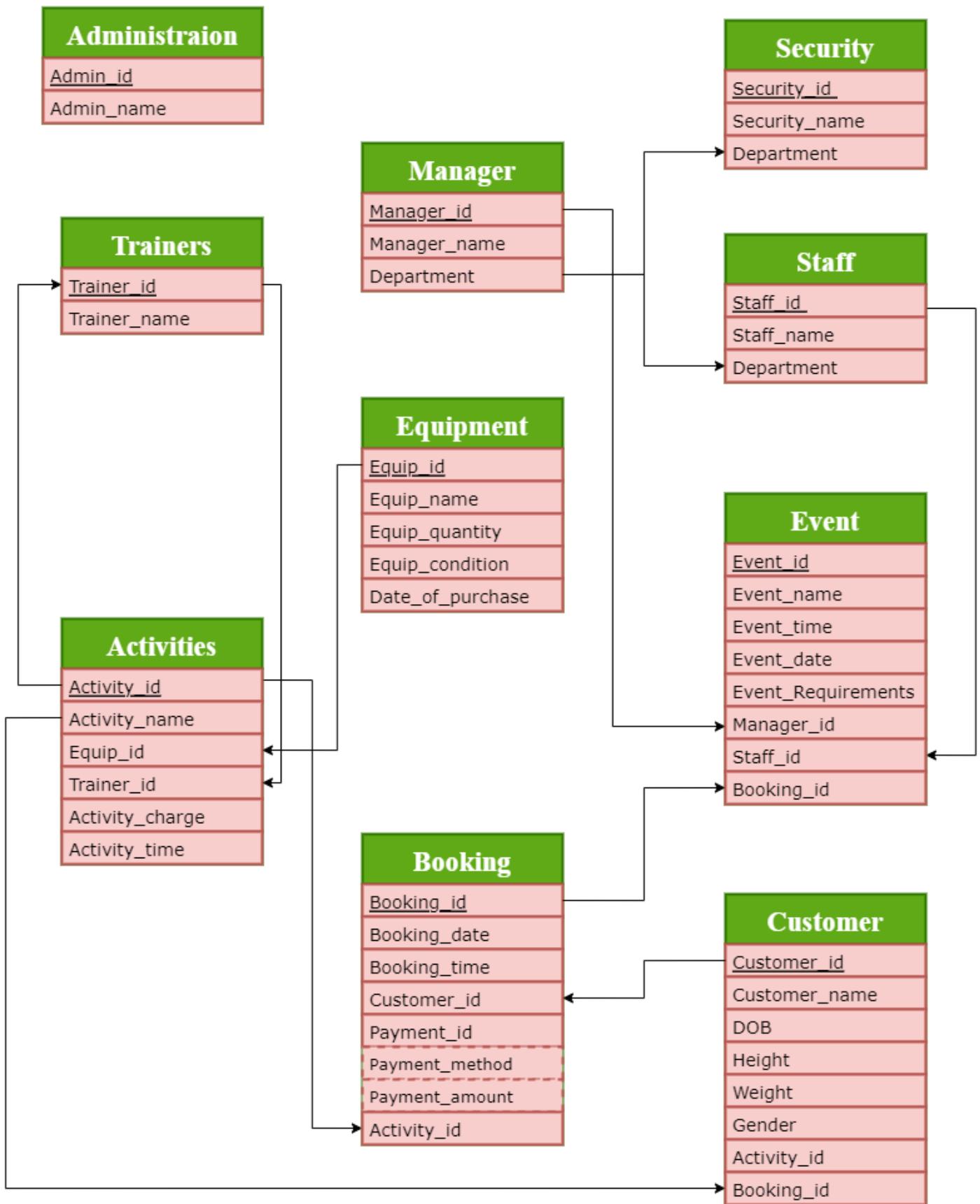
Since there is no transitive dependency, it is also in 3NF. Thus, our final schema remains the same.

For a relation to be in **BCNF**,

1. It should be in the third normal form (3NF).
2. For any dependency $A \rightarrow B$, A must be a super key.

Hence, this relation is in BCNF as well.

Updated Relational Model:



Final DDL and Select Statements

1. Administration (Admin_id, Admin_name)

---Administration

```
CREATE TABLE IF NOT EXISTS ba_ms."Administration"
(
    "Admin_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Admin_name" character(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Administration_pkey" PRIMARY KEY ("Admin_id")
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS ba_ms."Administration"
    OWNER to postgres;
```

```
COPY ba_ms."Administration"
FROM 'C:\Users\Public\sn_db\Administration.csv'
DELIMITER ',' CSV HEADER;
```

Query Query History

```
1 COPY ba_ms."Administration"
2 FROM 'C:\Users\Public\sn_db\Administration.csv'
3 DELIMITER ',' CSV HEADER;
4
5
6 SELECT * FROM ba_ms."Administration";
7
8
```

Loading...

Data output Messages Notifications

	Admin_id [PK] character varying	Admin_name character (50)
1	1	Mumnro
2	2	Judon
3	3	Willabella
4	4	Garrot
5	5	Cyrus
6	6	Malynda
7	7	Lou
8	8	Sofia
9	9	Vanny
10	10	Free

2. Manager (Manager_id, Manager_name, department)

--- Manager ---

```
CREATE TABLE IF NOT EXISTS ba_ms."Manager"
(
    "Manager_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Manager_name" character(50) COLLATE pg_catalog."default" NOT NULL,
    "Department" character(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Manager_id" PRIMARY KEY ("Manager_id")
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS ba_ms."Manager"
    OWNER to postgres;
```

```
COPY ba_ms."Manager"
FROM 'C:\Users\Public\sn_db\Manager.csv'
DELIMITER ',' CSV HEADER;
```

Query Query History

```
9
10 COPY ba_ms."Manager"
11 FROM 'C:\Users\Public\sn_db\Manager.csv'
12 DELIMITER ',' CSV HEADER;
13
14
15 SELECT * FROM ba_ms."Manager";
16
```

Data output Messages Notifications

The screenshot shows a PostgreSQL query interface with the following details:

- Query Tab:** The current tab is "Query".
- Code Area:** Displays the SQL code for creating the "Manager" table and copying data from a CSV file.
- Data Output Tab:** The "Data output" tab is selected.
- Table View:** A grid view of the "Manager" table data. The columns are "Manager_id", "Manager_name", and "Department". The data consists of 15 rows, each with a unique ID from 1 to 15, a name, and a department.
- Total Rows:** At the bottom left, it says "Total rows: 15 of 15".
- Completion Time:** At the bottom right, it says "Query complete 00:00:00.064".

	Manager_id	Manager_name	Department
1	1	Jo	Activity
2	2	Clay	Equipment
3	3	Reggis	Staff
4	4	Elfrieda	Security
5	5	Babita	Trainer
6	6	Paloma	Event
7	7	Aurlie	Booking
8	8	Georgina	Activity
9	9	Sam	Equipment
10	10	Dre	Staff
11	11	Dirk	Security
12	12	Gerrilee	Trainer
13	13	Elmira	Event
14	14	Annmaria	Booking

3. Staff (Staff_id, Staff_name, Department)

--- Staff ---

```
CREATE TABLE IF NOT EXISTS ba_ms."Staff"
(
    "Staff_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Staff_name" character(100) COLLATE pg_catalog."default" NOT NULL,
    "Department" character(100) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Staff_pkey" PRIMARY KEY ("Staff_id")
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS ba_ms."Staff"
    OWNER to postgres;

COPY ba_ms."Staff"
FROM 'C:\Users\Public\sn_db\Staff.csv'
DELIMITER ',' CSV HEADER;
```

Data output Messages Notifications

	Staff_id [PK] character varying	Staff_name character (100)	Department character (100)
1	1	Karlik	Activity
2	2	Melosa	Equipment
3	3	Ame	Staff
4	4	Dawn	Security
5	5	Francesco	Trainer
6	6	Reyna	Event
7	7	Barbe	Booking
8	8	Ivy	Activity
9	9	Tamiko	Equipment
10	10	Wally	Staff
11	11	Diann	Security
12	12	Michael	Trainer
13	13	Stanleigh	Event
14	14	Florance	Booking
15	15	Claudia	Activity
16	16	Mimi	Equipment
17	17	Christoper	Staff
18	18	Eden	Security
19	19	Elianore	Trainer
20	20	Jere	Event
21	21	Samaria	Booking

Total rows: 50 of 50 Query complete 00:00:00.070

4. Security (Security_id, Security_name, Department)

--- Security ---

```
CREATE TABLE IF NOT EXISTS bo_ms."Security"
(
    "Security_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Security_name" character(100) COLLATE pg_catalog."default" NOT NULL,
    "Department" character(100) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Security_pkey" PRIMARY KEY ("Security_id")
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS bo_ms."Security"
```

```
OWNER to postgres;
```

```
COPY ba_ms."Security"  
FROM 'C:\Users\Public\sn_db\Security.csv'  
DELIMITER ',' CSV HEADER;  
ACCESS
```

Query Query History

```
27 FROM 'C:\Users\Public\sn_db\Security.csv'  
28 DELIMITER ',' CSV HEADER;  
29  
30  
31 SELECT * FROM ba_ms."Security";
```

Data output Messages Notifications Loading...

	Security_id [PK] character varying	Security_name character (100)	Department character (100)
1	1	Nataline	Activity
2	2	Arlena	Equipment
3	3	Harland	Staff
4	4	Nicola	Security
5	5	Teddie	Trainer
6	6	Rickert	Event
7	7	Marius	Booking
8	8	Deonne	Activity
9	9	Biron	Equipment
10	10	Agnola	Staff
11	11	Melva	Security
12	12	Brinn	Trainer
13	13	Marcelo	Event
14	14	Gwynne	Booking
15	15	Sherlocke	Activity
16	16	Felicio	Equipment

Total rows: 50 of 50 Query complete 00:00:00.062

5. Customer (Customer_id, Customer_name, DOB, age, Height, Weight, Gender, Activtiy_id, Booking_id)

--- Customer ---

```
CREATE TABLE IF NOT EXISTS ba_ms."Customer"
(
    "Customer_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Customer_name" character(100) COLLATE pg_catalog."default" NOT NULL,
    "Gender" character(100) COLLATE pg_catalog."default" NOT NULL,
    "Activity_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "DOB" date NOT NULL,
    "Height" numeric NOT NULL,
    "Weight" numeric NOT NULL,
    CONSTRAINT "Customer_pkey" PRIMARY KEY ("Customer_id"),
    CONSTRAINT "Activity_id" FOREIGN KEY ("Activity_id")
        REFERENCES ba_ms."Activity" ("Activity_id") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS ba_ms."Customer"
    OWNER to postgres;

COPY ba_ms."Customer"
FROM 'C:\Users\Public\sn_db\Customer.csv'
DELIMITER ',' CSV HEADER;
```

Query Query History

```
72 COPY ba_ms."Customer"
73 FROM 'C:\Users\Public\sn_db\Customer.csv'
74 DELIMITER ',' CSV HEADER;
--
```

Data output Messages Notifications

	Customer_id [PK] character varying	Customer_name character (100)	Gender character (100)	Activity_id character varying	DOB date	Height numeric	Weight numeric
1	1	Pepillo	M	1	2003-07-09	163.07	84.84
2	2	Elise	F	2	2003-07-10	127.31	55.76
3	3	Raquela	F	3	2003-07-11	133.77	81.73
4	4	Martino	M	4	2003-07-12	154.11	101.24
5	5	Emmanuel	M	5	1990-12-11	157.57	75.65
6	6	Bordie	M	6	1990-12-12	104.37	112.1
7	7	Baudoin	M	7	2003-07-09	130.98	100.2
8	8	Hillel	M	8	1988-10-08	138.96	48.6
9	9	Glenda	F	9	1988-10-09	146.67	69.99
10	10	Lilly	F	10	1988-10-10	105.23	50.49
11	11	Cal	M	11	1988-10-11	113.79	91.99
12	12	Octavius	M	12	2000-04-08	119.33	113.92
13	13	Phillipe	M	13	2000-04-09	131.5	55.15
14	14	Maxwell	M	14	2000-04-10	165.64	53.78
15	15	Gare	M	15	2000-04-11	187.46	124.97
16	16	Jobye	F	1	2001-09-04	169.28	48.76
17	17	Abbe	M	2	2001-09-05	195.89	46.06

Total rows: 50 of 50 Query complete 00:00:00.058

6. Booking (Booking_id, Booking_date, Booking_time, Customer_id, Payment_id, Payment_method, Payment_amount, Activity_id)

--- Booking ---

```
CREATE TABLE IF NOT EXISTS ba_ms."Booking"
(
    "Booking_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Booking_date" date NOT NULL,
    "Booking_time" time without time zone NOT NULL,
    "Payment_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Payment_method" character(100) COLLATE pg_catalog."default" NOT NULL,
    "Payment_amount" numeric NOT NULL,
    "Activity_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Customer_id" character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Booking_pkey" PRIMARY KEY ("Booking_id"),
    CONSTRAINT "Activity_id" FOREIGN KEY ("Activity_id")
```

```

REFERENCES ba_ms."Activity" ("Activity_id") MATCH SIMPLE
ON UPDATE NO ACTION
ON DELETE NO ACTION
NOT VALID,
CONSTRAINT "Customer_id" FOREIGN KEY ("Customer_id")
REFERENCES ba_ms."Customer" ("Customer_id") MATCH SIMPLE
ON UPDATE NO ACTION
ON DELETE NO ACTION
NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

```

```

ALTER TABLE IF EXISTS ba_ms."Booking"
OWNER to postgres;

```

```

COPY ba_ms."Booking"
FROM 'C:\Users\Public\sn_db\Booking.csv'
DELIMITER ',' CSV HEADER;

```

Query Query History

```

34
35 COPY ba_ms."Booking"
36 FROM 'C:\Users\Public\sn_db\Booking.csv'
37 DELIMITER ',' CSV HEADER;

```

Data output Messages Notifications

	Booking_id [PK] character varying	Booking_date date	Booking_time time without time zone	Payment_id character varying	Payment_method character (100)	Payment_amount numeric	Activity_id character varying	Customer_id character varying
1	1	2023-05-01	00:37:00	201465000000000	Cash	6073	1	1
2	2	2023-05-02	08:02:00	5100150000000	Card	3169	2	2
3	3	2023-05-03	19:36:00	4405820000000	Upi	1736	3	3
4	4	2023-05-04	02:01:00	5602230000000	Cash	3204	4	4
5	5	2023-05-05	10:31:00	3581300000000	Card	3564	5	5
6	6	2023-05-06	16:28:00	5602240000000	Upi	8126	6	6
7	7	2023-05-07	18:43:00	3547000000000	Cash	2597	7	7
8	8	2023-05-08	08:40:00	3576590000000	Card	9213	8	8
9	9	2023-05-09	23:08:00	3569860000000	Upi	3914	9	9
10	10	2023-05-10	17:53:00	3533690000000	Cash	2443	10	10
11	11	2023-05-11	19:06:00	3057110000000	Card	3360	11	11
12	12	2023-05-12	10:44:00	5610670000000	Upi	6068	12	12
13	13	2023-05-13	05:54:00	3551460000000	Cash	5812	13	13
14	14	2023-05-14	19:15:00	5007670000000	Card	7198	14	14
15	15	2023-05-15	08:32:00	3567710000000	Upi	7780	15	15
16	16	2023-05-16	10:02:00	3550750000000	Cash	9715	1	16
17	17	2023-05-17	17:33:00	5602220000000	Card	2676	2	

✓ Table truncated

Total rows: 50 of 50 Query complete 00:00:00.073

7. Event (Event_id, Event_name, Event_time, Event_date, Event_requirements, manager_id, Staff_id, Booking_id)

--- Event ---

```
CREATE TABLE IF NOT EXISTS ba_ms."Event"
(
    "Event_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Event_name" character(100) COLLATE pg_catalog."default" NOT NULL,
    "Event_time" time without time zone NOT NULL,
    "Event_date" date NOT NULL,
    "Event_requirements" character varying COLLATE pg_catalog."default" NOT
NULL,
    "Manager_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Staff_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Booking_id" character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Event_pkey" PRIMARY KEY ("Event_id"),
    CONSTRAINT "Booking_id" FOREIGN KEY ("Booking_id")
        REFERENCES ba_ms."Booking" ("Booking_id") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Manager_id" FOREIGN KEY ("Manager_id")
        REFERENCES ba_ms."Manager" ("Manager_id") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Staff_id" FOREIGN KEY ("Staff_id")
        REFERENCES ba_ms."Staff" ("Staff_id") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS ba_ms."Event"
    OWNER to postgres;

COPY ba_ms."Event"
FROM 'C:\Users\Public\sn_db\Event.csv'
DELIMITER ',' CSV HEADER;
```

Query Query History

```
79
80 COPY ba_ms."Event"
81 FROM 'C:\Users\Public\sn_db\Event.csv'
82 DELIMITER ',' CSV HEADER;
```

Data output Messages Notifications

>Loading...

	Event_id [PK] character varying	Event_name character (100)	Event_time time without time zone	Event_date date	Event_requirements character varying	Manager_id character varying	Staff_id character varying	Booking_id character varying
1	1	Birthday	00:29:00	2023-06-02	balloons	1	1	1
2	2	Wedding	02:40:00	2024-03-11	lights	2	2	2
3	3	Office Party	12:52:00	2023-11-09	cardboards	3	3	3
4	4	Marriage Party	20:05:00	2024-10-01	cartoons	4	4	4
5	5	Anniversary	18:41:00	2023-11-11	chairs	5	5	5
6	6	Fest	16:14:00	2023-11-12	tables	6	6	6
7	7	College Events	13:57:00	2023-11-13	stage	7	7	7
8	8	Companies Eve...	06:23:00	2023-11-14	photographer	8	8	8
9	9	Birthday	15:38:00	2023-11-15	cook	9	9	9
10	10	Wedding	19:16:00	2023-01-05	candles	10	10	10
11	11	Office Party	05:48:00	2023-01-06	balloons	11	11	11
12	12	Marriage Party	19:22:00	2023-01-06	lights	12	12	12
13	13	Anniversary	20:43:00	2023-01-07	cardboards	13	13	13
14	14	Fest	07:33:00	2024-04-01	cartoons	14	14	14
15	15	College Events	21:23:00	2024-04-02	chairs	15	15	15
16	16	Companies Eve...	05:19:00	2024-04-03	tables	1	16	16
17	17	Birthday	07:16:00	2023-08-07	stage	2	17	17

Total rows: 50 of 50 Query complete 00:00:00.076

8. Trainers (Trainer_id, Trainer_name, Activity_id, activity_name)

--- Trainers —

```
CREATE TABLE IF NOT EXISTS ba_ms."Trainers"
(
    "Trainer_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Trainer_name" character(100) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Trainers_pkey" PRIMARY KEY ("Trainer_id")
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS ba_ms."Trainers"
OWNER to postgres;
```

```
COPY ba_ms."Trainers"
FROM 'C:\Users\Public\sn_db\Trainers.csv'
DELIMITER ',' CSV HEADER;
```

Query Query History

```
54 COPY ba_ms."Trainers"  
55 FROM 'C:\Users\Public\sn_db\Trainers.csv'  
56 DELIMITER ',' CSV HEADER;  
57  
58  
59 SELECT * FROM ba_ms."Trainers";
```

Data output Messages Notifications

	Trainer_id [PK] character varying	Trainer_name character (100)
1	1	Merline
2	2	Aube
3	3	Ursala
4	4	Iolanthe
5	5	Crystie
6	6	Rooney
7	7	Shena
8	8	Justis
9	9	Cammie
10	10	Wynn
11	11	Katalin
12	12	Osborne
13	13	Eachelle
14	14	Delmer
15	15	Emmalynn

Total rows: 15 of 15

Query complete 00:00:00.066

9. Activity (Activity_id, Activity_name, Activity_charge, Activity_time, Equip_id, Trainer_id)

--- Activity ---

```
CREATE TABLE IF NOT EXISTS ba_ms."Activity"  
(  
    "Activity_id" character varying COLLATE pg_catalog."default" NOT NULL,  
    "Activity_name" character(100) COLLATE pg_catalog."default" NOT NULL,  
    "Activity_charge(in $)" integer NOT NULL,  
    "Activity_time(in min)" integer NOT NULL,  
    "Equip_id" character varying COLLATE pg_catalog."default" NOT NULL,  
    "Trainer_id" character varying COLLATE pg_catalog."default" NOT NULL,  
    CONSTRAINT "Activity_pkey" PRIMARY KEY ("Activity_id"),  
    CONSTRAINT "Equip_id" FOREIGN KEY ("Equip_id")
```

```
REFERENCES ba_ms."Equipment" ("Equipment_id") MATCH SIMPLE
ON UPDATE NO ACTION
ON DELETE NO ACTION
NOT VALID,
CONSTRAINT "Trainer_id" FOREIGN KEY ("Trainer_id")
REFERENCES ba_ms."Trainers" ("Trainer_id") MATCH SIMPLE
ON UPDATE NO ACTION
ON DELETE NO ACTION
NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS bo_ms."Activity"
    OWNER to postgres;

COPY bo_ms."Activity"
FROM 'C:\Users\Public\sn_db\Activity.csv'
DELIMITER ',' CSV HEADER;
```

Query Query History

```
63 COPY ba_ms."Activity"  
64 FROM 'C:\Users\Public\sn_db\Activity.csv'  
65 DELIMITER ',' CSV HEADER;  
66  
67  
68 SELECT * FROM ba_ms."Activity";
```

Data output Messages Notifications

	Activity_id [PK] character varying	Activity_name character (100)	Activity_charge(in \$) integer	Activity_time(in min) integer	Equip_id character varying	Trainer_id character varying
1	1	Scuba Diving	17	30	1	1
2	2	Snorkeling	12	35	2	2
3	3	Paragliding	10	20	3	3
4	4	Parasailing	15	25	4	4
5	5	Flyboarding	18	40	5	5
6	6	Windsurfing	16	45	6	6
7	7	Beach Volleyball	14	50	7	7
8	8	Jet Ski	9	60	8	8
9	9	Kite surfing	15	120	9	9
10	10	Beach Sprint ro...	6	90	10	10
11	11	Banana Ride	16	30	11	11
12	12	Speed Boat	15	45	12	12
13	13	Dolphin Explora...	17	60	13	13
14	14	FootVolley	12	25	14	14
15	15	BeachCamping	18	40	15	15

Total rows: 15 of 15

Query complete 00:00:00.054

10. Equipment (Equip_id, Equip_name, Equip_quantity, Equip_condition, Date_of_purchase)

--- Equipment ---

```
CREATE TABLE IF NOT EXISTS ba_ms."Equipment"  
(  
    "Equipment_id" character varying COLLATE pg_catalog."default" NOT NULL,  
    "Equipment_name" character(100) COLLATE pg_catalog."default" NOT NULL,  
    "Equipment_quantity" numeric NOT NULL,  
    "Equipment_condition" character(100) COLLATE pg_catalog."default" NOT NULL,  
    "Date_of_purchase" date NOT NULL,  
    CONSTRAINT "Equipment_pkey" PRIMARY KEY ("Equipment_id")  
)  
WITH (  
    OIDS = FALSE  
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS ba_ms."Equipment"
OWNER to postgres;
```

```
COPY ba_ms."Equipment"
FROM 'C:\Users\Public\sn_db\Equipment.csv'
DELIMITER ',' CSV HEADER;
```

Query Query History

```
44 COPY ba_ms."Equipment"
45 FROM 'C:\Users\Public\sn_db\Equipment.csv'
46 DELIMITER ',' CSV HEADER;
47
48
49 SELECT * FROM ba_ms."Equipment";
```

Data output Messages Notifications

	Equipment_id [PK] character varying	Equipment_name character (100)	Equipment_quantity numeric	Equipment_condition character (100)	Date_of_purchase date
1	1	Life Jacket	66	new	2018-01-09
2	2	Helmet	54	old	2020-02-09
3	3	Boat	26	need to change	2017-03-09
4	4	Knives	60	new	2020-04-09
5	5	Hinges	16	old	2020-05-09
6	6	Ropes	31	need to change	2019-06-09
7	7	Parachutes	20	new	2020-07-09
8	8	Knee Gaurds	13	old	2020-08-09
9	9	Albow Gaurds	70	need to change	2020-09-09
10	10	Wistles	76	new	2020-10-09
11	11	Speed Boat	52	old	2020-11-09
12	12	Rowing Boats	33	need to change	2020-12-09
13	13	First Aid Kits	23	new	2018-01-19
14	14	Harness	39	old	2020-02-02
15	15	Lifebuoys	10	need to change	2017-03-05

Total rows: 30 of 30 Query complete 00:00:00.062

SQL Queries

Q1: To find all customer details whose name starts with ‘B’.

Ans:

--- Query 1 ---

```
SELECT *
FROM ba_ms."Customer"
WHERE "Customer_name" LIKE 'B%'
```

The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', and 'Queries.sql*'. Below the bar is a toolbar with various icons for file operations and database management. The main area is divided into two tabs: 'Query' (selected) and 'Query History'. The 'Query' tab contains the following code:

```
1 --- Query 1 ---
2 SELECT *
3 FROM ba_ms."Customer"
4 WHERE "Customer_name" LIKE 'B%'
5
6 --- Query 2 ---
```

Below the code, there are tabs for 'Data output', 'Messages', and 'Notifications'. The 'Data output' tab is selected and displays a table with the following data:

	Customer_id [PK] character varying	Customer_name character (100)	Gender character (100)	Activity_id character varying	DOB date	Height numeric	Weight numeric
1	6	Bordie	M	6	1990-12-12	104.37	112.1
2	7	Baudoin	M	7	2003-07-09	130.98	100.2
3	32	Bertine	F	2	1989-03-31	126.1	77.24

Q2: To find all customers whose height is greater than 150 cm.

Ans:

--- Query 2 ---

```
SELECT *
FROM ba_ms."Customer"
WHERE "Customer"."Height" > 150
ORDER BY "Customer"."Height"
```

The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a selected tab labeled 'Queries.sql'. Below the navigation bar is a toolbar with various icons for file operations, search, and database management. The main area is divided into two sections: 'Query' and 'Query History', with 'Query' being the active tab. The query text is displayed in the 'Query' section, starting with '--- Query 2 ---' and followed by the SELECT statement. The results of the query are shown in a table below, with 24 rows of data. The table has columns: Customer_id [PK] character varying, Customer_name character (100), Gender character (100), Activity_id character varying, DOB date, Height numeric, and Weight numeric. The 'Data output' tab is selected at the bottom, and the status bar at the bottom right indicates 'Query complete 00:00:00.090'.

	Customer_id [PK] character varying	Customer_name character (100)	Gender character (100)	Activity_id character varying	DOB date	Height numeric	Weight numeric
1	29	Piggy	M	14	1983-03-13	150.47	100.72
2	36	Titus	M	6	2005-11-04	152.25	91.51
3	4	Martino	M	4	2003-07-12	154.11	101.24
4	31	Ernestus	M	1	1989-03-30	154.55	69.72
5	42	Caria	F	12	1989-03-31	154.9	87.12
6	5	Emmanuel	M	5	1990-12-11	157.57	75.65
7	48	Reece	M	3	2004-12-15	158.32	127.55
8	41	Godard	M	11	1989-03-30	158.66	54.28
9	30	Rosemary	F	15	1983-03-14	158.66	123.9
10	49	Gar	M	4	1990-08-10	160.81	128.74
11	35	Zelda	F	5	2004-10-13	161.55	68.88
12	1	Pepillo	M	1	2003-07-09	163.07	84.84
13	34	Jodi	M	4	2004-10-12	164.3	63.46
14	14	Maxwell	M	14	2000-04-10	165.64	53.78

Total rows: 24 of 24 Query complete 00:00:00.090

Q3: To find all customers whose weight is less than 80 Kg.

Ans:

--- Query 3 ---

```
SELECT *
FROM ba_ms."Customer"
WHERE "Customer"."Weight" < 80
ORDER BY "Customer"."Weight"
```

The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a file named 'Queries.sql'. Below the navigation is a toolbar with various icons for file operations and search. The main area is divided into two sections: 'Query' and 'Query History'. The 'Query' section contains the SQL code for Q3. The 'Query History' section is currently empty. Below these is a 'Data output' tab, which is selected, showing a table of customer data. The table has columns: Customer_id [PK] character varying, Customer_name character (100), Gender character (100), Activity_id character varying, DOB date, Height numeric, and Weight numeric. The data shows 21 rows of customer information. At the bottom of the data output section, it says 'Total rows: 21 of 21' and 'Query complete 00:00:00.147'.

	Customer_id [PK] character varying	Customer_name character (100)	Gender character (100)	Activity_id character varying	DOB date	Height numeric	Weight numeric
1	17	Abbe	M	2	2001-09-05	195.89	46.06
2	8	Hillel	M	8	1988-10-08	138.96	48.6
3	16	Jobye	F	1	2001-09-04	169.28	48.76
4	10	Lilly	F	10	1988-10-10	105.23	50.49
5	44	Corey	F	14	1989-04-02	111.31	52.58
6	14	Maxwell	M	14	2000-04-10	165.64	53.78
7	22	Robenia	F	7	1996-08-08	101.76	53.81
8	41	Godard	M	11	1989-03-30	158.66	54.28
9	13	Phillipe	M	13	2000-04-09	131.5	55.15
10	2	Elise	F	2	2003-07-10	127.31	55.76
11	20	Tobye	F	5	1990-12-06	173.93	56.01
12	28	Ulla	F	13	1983-03-12	195.9	59.39
13	27	Hetti	F	12	1983-03-11	191.21	60.7
14	34	Jodi	M	4	2004-10-12	164.3	63.46

Q4: To find all managers who work in the security department.

Ans:

--- Query 4 ---

```
SELECT *
FROM ba_ms."Manager"
WHERE "Manager"."Department" LIKE 'Security%'
```

The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a file named 'Queries.sql'. Below the tabs is a toolbar with various icons for database management. The main area is divided into two sections: 'Query' and 'Query History'. The 'Query' section contains the SQL code for Q4. The 'Query History' section shows the previous query (Q3) and the current query (Q4). Below the query area is a 'Data output' tab, which is currently selected, showing a table with three columns: Manager_id, Manager_name, and Department. The table has two rows of data.

	Manager_id [PK] character varying	Manager_name character (50)	Department character (50)
1	4	Elfrieda	Security
2	11	Dirk	Security

Q5: Display Equipment_id and Equipment_name whose condition is old.

Ans:

--- Query 5 ---

```
SELECT ba_ms."Equipment"."Equipment_id", ba_ms."Equipment"."Equipment_name"
FROM ba_ms."Equipment"
WHERE "Equipment"."Equipment_condition" LIKE 'old%'
```

The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a selected tab labeled 'Queries.sql'. Below the tabs is a toolbar with icons for file operations, search, and execution. The main area contains the following code:

```
16 ORDER BY "Customer"."Weight"
17
18 --- Query 4 ---
19 SELECT *
20 FROM ba_ms."Manager"
21 WHERE "Manager"."Department" LIKE 'Security%'
22
```

Below the code, there are tabs for Data output, Messages, and Notifications. The Data output tab is selected, showing a table with three columns: Manager_id, Manager_name, and Department. The data is as follows:

	Manager_id [PK] character varying	Manager_name character (50)	Department character (50)
1	4	Elfrieda	Security
2	11	Dirk	Security

Q6: Display all the activity names corresponding to all the bookings made.

Ans:

--- Query 6 ---

```
SELECT ba_ms."Booking"."Booking_id", ba_ms."Booking"."Activity_id",
ba_ms."Activity"."Activity_name"
FROM ba_ms."Activity"
INNER JOIN ba_ms."Booking" ON ba_ms."Booking"."Activity_id" =
ba_ms."Activity"."Activity_id"
```

The screenshot shows the pgAdmin 4 interface. The top navigation bar includes 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', and 'Queries.sql'. The 'Queries.sql' tab is active, showing the SQL code for Query 6. Below the tabs is a toolbar with various icons. The main area displays the query results under the 'Data output' tab, which is also active. The results show two rows of data from the 'Manager' table, filtered by the 'Security' department.

	Manager_id [PK] character varying	Manager_name character (50)	Department character (50)
1	4	Elfrieda	Security
2	11	Dirk	Security

Q7: Trigger: If any customer is below the required age then they can't insert their data in customer details.

Ans:

--- Query 7 (Trigger) ---

```
CREATE FUNCTION ba_ms.Age()
RETURNS trigger
LANGUAGE 'plpgsql'
AS $BODY$
begin
if new."DOB">>'01-01-2006'
then
raise notice 'Age is low';
return null;
end if;
return new;
end;
$BODY$;
CREATE TRIGGER CheckUser
BEFORE INSERT
ON ba_ms."Customer"
FOR EACH ROW
EXECUTE procedure ba_ms.Age();
```

Dashboard Properties SQL Statistics Dependencies Dependents [Queries.sql*](#)

Section6_Group2_6.1/postgres@PostgreSQL 10

No limit

Query History

```

36
37
38 --- Query 7 (Trigger) ---
39
40 CREATE FUNCTION ba_ms.Age()
41 RETURNS trigger
42 LANGUAGE 'plpgsql'
43 AS $BODY$
44 begin
45 if new."DOB">>'01-01-2006'
46 then
47 raise notice 'Age is low';
48 return null;
49 end if;
50 return new;
51 end;
52 $BODY$;
53 CREATE TRIGGER CheckUser
54 BEFORE INSERT
55 ON ba_ms."Customer"
56 FOR EACH ROW
57 EXECUTE procedure ba_ms.Age();

```

Data output Messages Notifications

CREATE TRIGGER

Query returned successfully in 85 msec.

Total rows: 50 of 50 | Query complete 00:00:00.085

```
INSERT INTO ba_ms."Customer"
VALUES (52, 'Ronit', 'M', 6, '02-05-2008', 160, 67)
```

Dashboard Properties SQL Statistics Dependencies Dependents [Queries.sql*](#)

Section6_Group2_6.1/postgres@PostgreSQL 10

No limit

Query History

```

51 end;
52 $BODY$;
53 CREATE TRIGGER CheckUser
54 BEFORE INSERT
55 ON ba_ms."Customer"
56 FOR EACH ROW
57 EXECUTE procedure ba_ms.Age();

58
59
60 INSERT INTO ba_ms."Customer"
61 VALUES (52, 'Ronit', 'M', 6, '02-05-2008', 160, 67)
62
63

```

Data output Messages Notifications

NOTICE: Age is low

INSERT 0 0

Query returned successfully in 77 msec.

Q8: Function: In our function return the activity id in which the number of bookings is highest.

Ans:

```
create or replace function ba_ms."Find_Max"()
returns table(ActivityID character varying)
language 'plpgsql'
as $body$
declare
list_item record;
max_cnt integer;
begin
select max(T.count) into max_cnt from (select count(*) from ba_ms."Booking" group by
"Activity_id") as T;
create temp table Temp_Table(ActivityID character varying) on commit drop;
for list_item in (select "Activity_id" from (select "Activity_id",count(*) as cnt from
ba_ms."Booking" group by "Activity_id") as T
where T.cnt=max_cnt)
loop
insert into Temp_Table (ActivityId) values
(list_item."Activity_id");
end loop;
return query table Temp_Table;
end;
$body$;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a file tab labeled "Queries.sql*".
- Query Editor:** The main area displays the SQL code for the "Find_Max" function.
- Code Content:**

```
71 list_item record;
72 max_cnt integer;
73▼ begin
74 select max(T.count) into max_cnt from (select count(*) from ba_ms."Booking" group by "Activity_id") as T;
75 create temp table Temp_Table(ActivityID character varying) on commit drop;
76 for list_item in (select "Activity_id" from (select "Activity_id",count(*) as cnt from ba_ms."Booking" group by "Activity_id") as T
77 where T.cnt=max_cnt)
78▼ loop
79 insert into Temp_Table (ActivityId) values
80 (list_item."Activity_id");
81 end loop;
82 return query table Temp_Table;
83 end;
84 $body$;
85 select * from ba_ms."Find_Max"();
86
```
- Data Output:** Shows the message "CREATE FUNCTION" and "Query returned successfully in 84 msec."

```
select * from bo_ms."Find_Max"();
```

The screenshot shows a PostgreSQL query editor interface. The top menu bar includes Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a tab labeled "Queries.sql*". Below the menu is a toolbar with various icons for database management. The main area displays a PL/pgSQL script named "Queries.sql*". The script contains several lines of code, including a loop that inserts activity IDs into a temporary table and a final SELECT statement. A comment "-- Query 10 ---" is present between two SELECT statements. The bottom section of the interface shows the results of the last query, which is a table with one column named "activityid" containing the values 2, 4, 3, 5, and 1.

```
76 for list_item in (select "Activity_id" from (select "Activity_id",count(*) as cnt from ba_ms."Booking" group by "Activity_id") as T
77     where T.cnt=max_cnt)
78 loop
79     insert into Temp_Table (ActivityId) values
80     (list_item."Activity_id");
81 end loop;
82 return query table Temp_Table;
83 end;
84 $body$;
85 select * from ba_ms."Find_Max"();
86
87 select * from (select "Activity_id",count(*) as cnt from ba_ms."Booking" group by "Activity_id") as T where T.cnt = (select * from T
88
89
90 --- Query 10 ---
91 SELECT *
```

Data output Messages Notifications

activityid
2
4
3
5
1

Q9: Print names of trainers associated with corresponding activity.

Ans:

--- Query 9 ---

```
SELECT ba_ms."Trainers"."Trainer_id", ba_ms."Trainers"."Trainer_name",
ba_ms."Activity"."Activity_id"
FROM ba_ms."Activity"
JOIN ba_ms."Trainers"
ON ba_ms."Activity"."Trainer_id" = ba_ms."Trainers"."Trainer_id"
```

The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and the active tab, Queries.sql*. Below the tabs is a toolbar with various icons for file operations, search, and execution. The main area is divided into sections: Query, Query History, Data output, and Messages/Notifications. The Query section contains the SQL code for Q9. The Data output section displays a table with 15 rows of data, which is the result of the query. The table has three columns: Trainer_id, Trainer_name, and Activity_id. The footer shows a message indicating 15 total rows and a completion time of 00:00:00.102.

	Trainer_id character varying	Trainer_name character (100)	Activity_id character varying
1	1	Merline	1
2	2	Aube	2
3	3	Ursala	3
4	4	Iolanthe	4
5	5	Crystie	5
6	6	Rooney	6
7	7	Shena	7
8	8	Justis	8
9	9	Cammie	9
10	10	Wynn	10
11	11	Katalin	11
12	12	Osborne	12
13	13	Eachelle	13
14	14	Delmer	14
15	15	Emmalynn	15

Total rows: 15 of 15 | Query complete 00:00:00.102

Q10: Print manager name associated with event id.

Ans:

--- Query 10 ---

```
SELECT ba_ms."Event"."Event_id",ba_ms."Manager"."Manager_name"
FROM ba_ms."Event"
JOIN ba_ms."Manager"
ON ba_ms."Event"."Manager_id"=ba_ms."Manager"."Manager_id"
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a tab labeled "Queries.sql*" which is currently selected.
- Connection Bar:** Shows the connection as "Section6_Group2_6.1/postgres@PostgreSQL 10".
- Query Editor:** Displays the SQL query with line numbers 88 through 93. Lines 88 and 89 are comments, while lines 90 through 93 form the SELECT statement.
- Data Output:** A table showing the results of the query. The columns are "Event_id" and "Manager_name". The data consists of 14 rows, each with an Event_id and a corresponding Manager_name.

	Event_id	Manager_name
1	46	Jo
2	31	Jo
3	16	Jo
4	1	Jo
5	47	Clay
6	32	Clay
7	17	Clay
8	2	Clay
9	48	Reggis
10	33	Reggis
11	18	Reggis
12	3	Reggis
13	49	Elfrieda
14	34	Elfrieda

- Status Bar:** Shows "Total rows: 50 of 50" and "Query complete 00:00:00.076".

Q11: Print names of staff members from equipment.

Ans:

--- Query 11 ---

```
Select *  
FROM ba_ms."Staff"  
WHERE ba_ms."Staff"."Department" Like 'Equipment%';
```

The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', and 'Queries.sql*'. Below the bar are various toolbar icons for file operations, search, and execution. The main area contains the SQL code for Query 11, which selects all columns from the 'Staff' table where the department starts with 'Equipment'. The code is numbered from 98 to 107. Below the code, there are tabs for 'Data output', 'Messages', and 'Notifications'. The 'Data output' tab is selected, displaying a table with three columns: 'Staff_id', 'Staff_name', and 'Department'. The data shows seven rows of staff members, all belonging to the 'Equipment' department. At the bottom of the interface, a status bar indicates 'Total rows: 7 of 7' and 'Query complete 00:00:00.088'.

	Staff_id	Staff_name	Department
1	2	Melosa	Equipment
2	9	Tamiko	Equipment
3	16	Mimi	Equipment
4	23	Danika	Equipment
5	30	Georg	Equipment
6	37	Emmie	Equipment
7	44	Ky	Equipment

Q12: Print activity name whose activity time is greater or equal to 60 min.

Ans:

--- Query 12 ---

```
SELECT ba_ms."Activity"."Activity_name", ba_ms."Activity"."Activity_time(in min)"
FROM ba_ms."Activity"
WHERE ba_ms."Activity"."Activity_time(in min)" >= 60
```

The screenshot shows a PostgreSQL client interface with the following details:

- Toolbar:** Includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a file icon for Queries.sql*.
- Connection:** Section6_Group2_6.1/postgres@PostgreSQL 10.
- Query Bar:** Contains icons for file, database, search, and other functions, along with a dropdown for No limit and a refresh button.
- Query List:** Shows numbered lines of code:
 - 98: JOIN ba_ms."Trainers"
 - 99: ON ba_ms."Activity"."Trainer_id" = ba_ms."Trainers"."Trainer_id"
 - 100:
 - 101:
 - 102: --- Query 11 ---
 - 103: Select *
 - 104: FROM ba_ms."Staff"
 - 105: WHERE ba_ms."Staff"."Department" Like 'Equipment%';
 - 106:
 - 107: --- Query 12 ---
 - 108: SELECT ba_ms."Activity"."Activity_name", ba_ms."Activity"."Activity_time(in min)"
 - 109: FROM ba_ms."Activity"
 - 110: WHERE ba_ms."Activity"."Activity_time(in min)" >= 60
 - 111:
- Data Output:** Shows a table with columns Activity_name and Activity_time(in min). The data is:

	Activity_name	Activity_time(in min)
1	Jet Ski	60
2	Kite surfing	120
3	Beach Sprint ro...	90
4	Dolphin Explora...	60
- Status Bar:** Total rows: 4 of 4 | Query complete 00:00:00.130

Q13: Display the details of customers with booking IDs greater than 25 who have done payments by UPI.

Ans:

--- Query 13 ---

```
SELECT *
FROM ba_ms."Customer"
WHERE ba_ms."Customer"."Customer_id" IN
  (SELECT ba_ms."Booking"."Customer_id"
   FROM ba_ms."Booking"
   WHERE ba_ms."Booking"."Payment_method" LIKE 'Upi%'
   AND ba_ms."Booking"."Booking_id" > '25')
```

The screenshot shows a database interface with the following components:

- Top Bar:** Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, [Queries.sql*](#).
- Toolbar:** Includes icons for file operations (New, Open, Save, Delete), search, and various query options.
- Section Header:** Section6_Group2_6.1/postgres@PostgreSQL 10.
- Query Tab:** Active tab, showing the SQL code for Query 13.
- Data Output Tab:** Active tab, displaying the results of the query.
- Results Table:** Shows 11 rows of customer data with columns: Customer_id, Customer_name, Gender, Activity_id, DOB, Height, and Weight.
- Bottom Status Bar:** Total rows: 11 of 11, Query complete 00:00:00.082.

	Customer_id [PK] character varying	Customer_name character (100)	Gender character (100)	Activity_id character varying	DOB date	Height numeric	Weight numeric
1	3	Raquela	F	3	2003-07-11	133.77	81.73
2	6	Bordie	M	6	1990-12-12	104.37	112.1
3	9	Glenda	F	9	1988-10-09	146.67	69.99
4	27	Hetti	F	12	1983-03-11	191.21	60.7
5	30	Rosemary	F	15	1983-03-14	158.66	123.9
6	33	Corabel	F	3	1989-04-01	178.12	68.77
7	36	Titus	M	6	2005-11-04	152.25	91.51
8	41	Godard	M	11	1989-03-30	158.66	54.28
9	44	Corey	F	14	1989-04-02	111.31	52.58
10	47	Dacy	F	2	2004-12-14	143.59	87.49
11	50	Shane	M	5	1990-08-11	131.82	94.11

Q14: Display booking ID and customer ID for an activity ID = 5

Ans:

```
SELECT
ba_ms."Booking"."Booking_id",ba_ms."Booking"."Customer_id",ba_ms."Activity"."Activity_id"
FROM ba_ms."Booking"
JOIN ba_ms."Customer"
ON ba_ms."Booking"."Customer_id"=ba_ms."Customer"."Customer_id"
JOIN ba_ms."Activity"
ON ba_ms."Booking"."Activity_id"=ba_ms."Activity"."Activity_id"
WHERE ba_ms."Activity"."Activity_id" = '5'
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and the current tab, `Queries.sql*`.
- Query Editor:** Shows the SQL code for Query 14, which selects booking ID, customer ID, and activity ID where the activity ID is 5.
- Data Output:** A table showing the results of the query execution. The columns are `Booking_id`, `Customer_id`, and `Activity_id`. The data consists of four rows with values (5, 5), (20, 20), (35, 35), and (50, 50).
- Status Bar:** At the bottom, it displays "Total rows: 4 of 4" and "Query complete 00:00:00.081".

Q15: Find booking date of female customer whose dob is greater than 01 - 01 - 2000 ?

Ans:

--- Query 15 ---

```
Select ba_ms."Booking"."Booking_date"
FROM ba_ms."Booking"
INNER JOIN ba_ms."Customer" ON ba_ms."Booking"."Customer_id" =
ba_ms."Customer"."Customer_id"
WHERE ba_ms."Customer"."Gender" like 'F%' and ba_ms."Customer"."DOB" > '01-01-2000';
```

The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', and a tab labeled 'Queries.sql*'. Below the navigation is a toolbar with various icons for file operations and database management. The main area is divided into sections: 'Query' (selected), 'Query History', and 'Data output'. The 'Query' section contains the SQL code for Question 15. The 'Data output' section displays a table with 8 rows, each containing a booking date. The bottom status bar indicates 'Total rows: 8 of 8' and 'Query complete 00:00:00.075'.

	Booking_date
1	2023-05-02
2	2023-05-03
3	2023-05-16
4	2023-05-23
5	2023-06-04
6	2023-06-07
7	2023-06-08
8	2023-06-16

Total rows: 8 of 8 Query complete 00:00:00.075

Q16: Print the top 5 admin names.

Ans:

--- Query 16 ---

```
Select *
FROM ba_ms."Administration" limit 5;
```

The screenshot shows a PostgreSQL client interface with the following details:

- Toolbar:** Includes Dashboard, Properties, SQL, Statistics, Dependencies, and a search/filter icon.
- Section Header:** Section6_Group2_6.1/postgres@PostgreSQL 10
- Query Tab:** Active tab, showing the executed query.
- Query Content:** The query text is displayed, starting with line 138: `--- Query 16 ---`, followed by `Select *`, `FROM ba_ms."Administration" limit 5;`, and line 141.
- Data Output:** A table showing the results of the query. The table has two columns: `Admin_id` and `Admin_name`. The data is as follows:

	Admin_id	Admin_name
1	1	Munmro
2	2	Judon
3	3	Willabella
4	4	Garrot
5	5	Cyrus

Q17: Print details of all the birthday events.

Ans:

--- Query 17 ---

```
Select *
FROM ba_ms."Event"
WHERE ba_ms."Event"."Event_name" Like 'Birthday%';
```

The screenshot shows a PostgreSQL database interface with the following details:

- Toolbar:** Includes Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a tab labeled "Queries.sql*".
- Query Editor:** Shows the code for Query 17, which selects all columns from the "Event" table where the "Event_name" is like 'Birthday%'.
- Data Output:** A table displaying the results of the query. The table has the following structure and data:

	Event_id [PK] character varying	Event_name character (100)	Event_time time without time zone	Event_date date	Event_requirements character varying	Manager_id character varying	Staff_id character varying	Booking_id character varying
1	1	Birthday	00:29:00	2023-06-02	balloons	1	1	1
2	9	Birthday	15:38:00	2023-11-15	cook	9	9	9
3	17	Birthday	07:16:00	2023-08-07	stage	2	17	17
4	25	Birthday	14:33:00	2023-11-09	chairs	10	25	25
5	33	Birthday	02:16:00	2023-04-10	cardboards	3	33	33
6	41	Birthday	11:53:00	2024-06-06	balloons	11	41	41
7	49	Birthday	13:20:00	2024-09-12	cook	4	49	49

Q18: Find staff_name whose event requirement is a photographer.

Ans:

--- Query 18 ---

```
Select ba_ms."Staff"."Staff_name"
FROM ba_ms."Staff"
INNER JOIN ba_ms."Event" ON ba_ms."Event"."Staff_id" = ba_ms."Staff"."Staff_id"
WHERE ba_ms."Event"."Event_requirements" LIKE 'photographer%';
```

The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', and 'Queries.sql*'. Below the bar is a toolbar with various icons for file operations and database management. The main area is divided into two tabs: 'Query' (selected) and 'Query History'. The 'Query' tab contains the SQL code for Query 18, which selects staff names from the 'Staff' table where their event requirements include the word 'photographer'. The code is numbered 148 through 153. Below the code, there are tabs for 'Data output', 'Messages', and 'Notifications'. The 'Data output' tab is selected and displays a table with one column, 'Staff_name', containing five rows: Ivy, Eden, Hilary, Winifred, and Dennie.

	Staff_name
1	Ivy
2	Eden
3	Hilary
4	Winifred
5	Dennie

Q19: Find the customer_id of booking done more than \$15.

Ans:

-- Query 19 --

```
Select ba_ms."Customer"."Customer_id"
FROM ba_ms."Booking"
INNER JOIN ba_ms."Customer" ON ba_ms."Booking"."Customer_id" =
ba_ms."Customer"."Customer_id"

INNER JOIN ba_ms."Activity" ON ba_ms."Booking"."Activity_id" =
ba_ms."Activity"."Activity_id"
WHERE ba_ms."Activity"."Activity_charge(in $)" > 15;
```

The screenshot shows the DBeaver SQL editor interface. The top navigation bar includes Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a tab labeled "Queries.sql" which is currently selected. Below the navigation bar is a toolbar with various icons for database management tasks. The main workspace is divided into two sections: "Query" and "Query History". The "Query" section contains the SQL code for Question 19, starting with a comment "-- Query 19 --" and selecting customer IDs from the Booking table based on activity charges. The "Data output" tab is active, displaying a table with the results of the query. The table has one column named "Customer_id" with the following data:

	Customer_id
1	1
2	5
3	6
4	11
5	13
6	15
7	16
8	20
9	21
10	26
11	28
12	30
13	31

At the bottom of the interface, a status bar indicates "Total rows: 20 of 20" and "Query complete 00:00:00.213".

Q20: Print manager_id, staff_id where the department is activity.

Ans:

--- Query 20 ---

```
SELECT ba_ms."Manager"."Manager_id", ba_ms."Staff"."Staff_id"
FROM ba_ms."Manager"
JOIN ba_ms."Staff"
ON ba_ms."Manager"."Department"=ba_ms."Staff"."Department"
WHERE ba_ms."Manager"."Department" LIKE 'Activity%'
```

The screenshot shows a PostgreSQL client interface with the following details:

- Toolbar:** Includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and a file icon labeled "Queries.sql*".
- Connection:** Set to "Section6_Group2_6.1/postgres@PostgreSQL 10".
- Query Bar:** Contains icons for folder, search, refresh, and various query options.
- Query History:** Shows the current query number (163) and the query itself:

```
163
164 --- Query 20 ---
165 SELECT ba_ms."Manager"."Manager_id", ba_ms."Staff"."Staff_id"
166 FROM ba_ms."Manager"
167 JOIN ba_ms."Staff"
168 ON ba_ms."Manager"."Department"=ba_ms."Staff"."Department"
169 WHERE ba_ms."Manager"."Department" LIKE 'Activity%'
```
- Data Output:** Shows a table with two columns: Manager_id and Staff_id. The data is as follows:

	Manager_id	Staff_id
1	15	1
2	8	1
3	1	1
4	15	8
5	8	8
6	1	8
7	15	15
8	8	15
9	1	15
10	15	22
11	8	22

- Metrics:** Total rows: 24 of 24 | Query complete 00:00:00.063