ΣΤΕΛΙΟΣ ΒΑΚΟΥΦΗΣ

Τρίτη Σειρά Ασκήσεων SQL

1. Εμφανίστε τα στοιχεία των πτήσεων με ημερομηνία αναχώρησης την 01/05/2018 και προορισμό το Τορόντο.

```
SELECT *
	FROM flights AS f
	WHERE f.depDate = '2018/05/01' AND
		f.toCity = 'Τορόντο'
;
```

2. Εμφανίστε έναν κατάλογο με τα στοιχεία των πτήσεων των οποίων η απόσταση κυμαίνεται μεταξύ 900 και 1500 μιλίων. Ο κατάλογος θα πρέπει να είναι ταξινομημένος με βάση την απόσταση σε αύξουσα διάταξη.

```
SELECT *
	FROM flights AS f
	WHERE f.distance BETWEEN 900 AND 1500
	ORDER BY f.distance ASC
;
```

3. Εμφανίστε έναν κατάλογο με τον συνολικό αριθμό των πτήσεων με ημερομηνία αναχώρησης μεταξύ 1/5/2018 μέχρι και 30/5/2018 ανά προορισμό.

```
SELECT f.toCity, COUNT(*) AS "Flights"
	FROM flights AS f
	WHERE f.depDate BETWEEN '2018/5/1' AND '2018/5/30'
	GROUP BY f.toCity
;
```

4. Εμφανίστε έναν κατάλογο με τους προορισμούς και τον συνολικό αριθμό των πτήσεων ανά προορισμό. Στον κατάλογο θα πρέπει να εμφανίζονται μόνο οι προορισμοί για τους οποίους υπάρχουν τουλάχιστον τρείς πτήσεις.

```
SELECT f.toCity, COUNT(*) AS "Flights"
	FROM flights AS f
	GROUP BY f.toCity
	HAVING COUNT(*) >= 3
;
```

5. Εμφανίστε έναν κατάλογο με το ονοματεπώνυμο των πιλότων που είναι πιστοποιημένοι στην λειτουργία τουλάχιστον τριών αεροσκαφών.

```
SELECT e.firstname, e.lastname
	FROM employees AS e
	WHERE empid = ANY(
		SELECT c.empid
			FROM certified AS c
			GROUP BY c.empid
			HAVING COUNT(*) >= 3
	)
;
```

6. Εμφανίστε το συνολικό κόστος των μηνιαίων μισθών όλων των υπαλλήλων της εταιρείας.

```
SELECT SUM(e.salary) AS "Employees Monthly Salary"
	FROM employees AS e
;
```

7. Εμφανίστε το συνολικό κόστος των μηνιαίων μισθών όλων των πιλότων της εταιρείας.

```
SELECT SUM(e.salary) AS "Pilots Monthly Salary"
	FROM employees AS e
	WHERE e.empid = ANY(
		SELECT DISTINCT c.empid
			FROM certified AS c
	)
;
```

8. Εμφανίστε το συνολικό κόστος των μηνιαίων μισθών των υπαλλήλων της εταιρείας που δεν είναι πιλότοι.

```
SELECT SUM(e.salary) AS "Not Pilots Monthly Salary"
	FROM employees AS e
	WHERE e.empid != ALL(
		SELECT DISTINCT c.empid
			FROM certified AS c
	)
;
```

9. Εμφανίστε έναν κατάλογο με τα ονόματα των αεροσκαφών που μπορούν να καλύψουν την πτήση από Αθήνα προς Μελβούρνη δίχως στάση για ανεφοδιασμό.

```
SELECT aname
	FROM aircrafts
	WHERE crange >= (
		SELECT f.distance
			FROM flights AS f
			WHERE f.fromCity = 'Αθήνα' AND
				f.toCity = 'Μελβούρνη'
	)
;
```

10. Εμφανίστε το ονοματεπώνυμο των πιλότων που είναι πιστοποιημένοι στην λειτουργία κάποιου αεροσκάφους τύπου Boeing (το όνομα του αεροσκάφους ξεκινάει με Boeing).

```
SELECT e.firstname, e.lastname
	FROM employees AS e
	WHERE e.empid = ANY(
		SELECT c.empid
			FROM certified AS c
			INNER JOIN aircrafts AS a
				ON c.aid = a.aid
			WHERE a.aname LIKE 'Boeing%'
	)
;
```

11. Βρείτε το ονοματεπώνυμο των πιλότων που είναι πιστοποιημένοι σε αεροσκάφη με δυνατότητα πτήσης μεγαλύτερης των 3000 μιλίων, αλλά δεν είναι πιστοποιημένοι σε κανένα αεροσκάφος τύπου Boeing.

```
SELECT e.firstname, e.lastname, e.empid
	FROM employees AS e
	WHERE e.empid = ANY (
		SELECT c.empid
			FROM certified AS c
			INNER JOIN aircrafts AS a
				ON c.aid = a.aid
			GROUP BY c.empid
			HAVING (SUM(CASE WHEN a.aname LIKE '%Boeing%' THEN 1 ELSE 0 END) = 0) AND
					(SUM(CASE WHEN a.crange >= 3000 THEN 1 ELSE 0 END) > 0)
	)
;
```

12. Βρείτε το ονοματεπώνυμο των υπαλλήλων με τον υψηλότερο μισθό.

```
SELECT e.firstname, e.lastname
	FROM employees AS e
	WHERE e.salary = (
		SELECT MAX(e.salary)
		FROM employees AS e
	)
;
```

13. Βρείτε το ονοματεπώνυμο των υπαλλήλων που έχουν τον δεύτερο υψηλότερο μισθό.

```
SELECT e.firstname, e.lastname
	FROM employees AS e
	WHERE e.salary = (
		SELECT salary
			FROM (
			SELECT
				salary, ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_num
				FROM (SELECT DISTINCT salary FROM employees) AS dist_sal
			) AS ar
		WHERE ar.row_num = 2
	)
;
```

14. Βρείτε τα ονόματα των αεροσκαφών για τα οποία όλοι οι πιστοποιημένοι στην λειτουργία τους πιλότοι έχουν μισθό τουλάχιστον 6000 ευρώ.

```
SELECT a.aname
	FROM aircrafts AS a
	INNER JOIN certified AS c
		ON a.aid = c.aid
	INNER JOIN employees AS e
		ON c.empid = e.empid
	GROUP BY a.aname
	HAVING (SUM(CASE WHEN e.salary >= 6000 THEN 1 ELSE 0 END) = 1) AND
			(SUM(CASE WHEN e.salary < 6000 THEN 1 ELSE 0 END) = 0)
;
```

15. Για κάθε πιλότο που είναι πιστοποιημένος στην λειτουργία τουλάχιστον τριών αεροσκαφών, βρείτε τον κωδικό του και το μεγαλύτερο crange των αεροσκαφών στα οποία είναι πιστοποιημένος.

```
SELECT e.empid, MAX(a.crange) AS "Max crange"
       FROM employees AS e
       INNER JOIN certified AS c
              ON e.empid = c.empid
       INNER JOIN aircrafts AS a
              ON c.aid = a.aid
       GROUP BY e.empid
       HAVING COUNT(*) >= 3
;
```

16. Βρείτε το ονοματεπώνυμο των υπαλλήλων με μισθό μικρότερο από το κόστος της φθηνότερη πτήσης με προορισμό την Μελβούρνη.

```
SELECT e.firstname, e.lastname
       FROM employees AS e
       WHERE e.salary < (
              SELECT MIN(f.price)
                     FROM flights AS f
                     WHERE f.toCity = 'Μελβούρνη'
       )
;
```

17. Βρείτε το ονοματεπώνυμο και τον μισθό των υπαλλήλων που δεν είναι πιλότοι και κερδίζουν πάνω από τον μέσο όρο του μισθού των πιλότων.

```
SELECT DISTINCT e.firstname, e.lastname, e.salary
       FROM employees AS e, certified AS c
              WHERE e.salary > (
                     SELECT AVG(e.salary)
                            FROM employees AS e
                            WHERE e.empid = ANY(
                                   SELECT DISTINCT c.empid
                                          FROM certified AS c
                            )
              ) AND e.empid != ALL (SELECT DISTINCT empid FROM certified)
;
```

18. Δημιουργείστε δύο όψεις. Η πρώτη όψη (pilots) θα περιέχει όλα τα στοιχεία των πιλότων και η δεύτερη (others) θα περιέχει όλα τα στοιχεία των υπαλλήλων που δεν είναι πιλότοι. Χρησιμοποιώντας τις όψεις που δημιουργήσατε και ξαναγράψτε τα ερωτήματα 7, 8 και 17.

```
CREATE VIEW pilots AS
(
	SELECT *
		FROM employees AS e
		WHERE e.empid = ANY(
			SELECT DISTINCT c.empid
				FROM certified AS c
		)
);

CREATE VIEW others AS
(
	SELECT *
		FROM employees AS e
		WHERE e.empid != ALL(
			SELECT DISTINCT c.empid
				FROM certified AS c
		)
);


/*QUERY 7*/
SELECT SUM(p.salary) AS "Pilots Monthly Salary"
	FROM pilots AS p
;

/*QUERY 8*/
SELECT SUM(o.salary) AS "Not Pilots Monthly Salary"
	FROM others AS o
;

/*QUERY 17*/
SELECT o.firstname, o.lastname, o.salary
	FROM others AS o
	WHERE o.salary > (SELECT AVG(salary) FROM pilots)
;
```

19. Δημιουργείστε μία όψη η οποία θα περιέχει το όνομα κάθε αεροσκάφους και τα στοιχεία των πτήσεων (fno, fromCity, toCity) που το κάθε αεροσκάφος μπορεί να καλύψει δίχως ανεφοδιασμό. Χρησιμοποιώντας την όψη που δημιουργήσατε εμφανίστε έναν κατάλογο με τα ονόματα των αεροσκαφών και τον αριθμό των πτήσεων που κάθε αεροσκάφος μπορεί να εξυπηρετήσει.

```
CREATE VIEW flight_aircrafts AS
	SELECT a.aname, f.fno, f.fromCity, f.toCity
		FROM flights AS f
		LEFT JOIN aircrafts AS a
			ON f.distance <= a.crange
;


SELECT fa.aname, COUNT(fa.fno) AS "Services"
	FROM flight_aircrafts AS fa
	GROUP BY fa.aname
;
```

20. Δημιουργείστε μια διαδικασία η οποία θα εμφανίζει τον κωδικό κάθε πτήσης και δίπλα τον χαρακτηρισμό "Φθηνή", "Κανονική" ή "Ακριβή". Μία πτήση θεωρείται φθηνή αν το κόστος του εισιτηρίου είναι μέχρι και 500 ευρώ, κανονική αν το κόστος κυμαίνεται μεταξύ 501 και 1500 ευρώ και ακριβή αν το κόστος του εισιτηρίου ξεπερνάει τα 1500 ευρώ.

```
CREATE PROCEDURE flight_verbal_cost
       AS
              SELECT f.fno,
                     CASE
                            WHEN f.price <= 500 THEN 'Cheap'
                            WHEN f.price <= 1500 THEN 'Normal' ELSE 'EXPENSIVE'
                     END AS cost
                     FROM flights AS f
;


/*Procedure Execution*/
EXEC flight_verbal_cost;
```

21. Δημιουργείστε μια διαδικασία η οποία θα δέχεται ως παραμέτρους το όνομα και τον κωδικό ενός πιλότου καθώς επίσης και το όνομα και τον κωδικό ενός αεροσκάφους. Η διαδικασία θα πιστοποιεί τον πιλότο στο συγκεκριμένο αεροσκάφος. Αν ο πιλότος ή το αεροσκάφος δεν υπάρχουν στην βάση δεδομένων η διαδικασία θα πρέπει να τα εισαγάγει. Σε περίπτωση που ο πιλότος είναι ήδη πιστοποιημένος στην λειτουργία του συγκεκριμένου αεροσκάφους η διαδικασία θα εμφανίζει κατάλληλο μήνυμα.

```
CREATE PROCEDURE insert_pilot
	(
	@pilot_id INT, @pilot_lname VARCHAR(30), @pilot_fname VARCHAR(30),
	@aircraft_id INT, @aircraft_name VARCHAR(50)
	)

	AS
		IF (EXISTS (SELECT empid FROM certified WHERE empid = @pilot_id AND aid = @aircraft_id))
			BEGIN
				PRINT 'Pilot already registered with this aircraft.'
				RETURN
			END

		DECLARE @pilot_updated BIT
		DECLARE @aircraft_updated BIT
		SET @pilot_updated = 1;
		SET @aircraft_updated = 1;

		IF (NOT EXISTS (SELECT empid FROM pilots WHERE empid = @pilot_id))
			BEGIN
				INSERT INTO employees (empid, firstname, lastname)
					VALUES (@pilot_id, @pilot_fname, @pilot_lname)
			END
		ELSE SET @pilot_updated = 0;

		IF (NOT EXISTS (SELECT aid FROM aircrafts WHERE aid = @aircraft_id))
			BEGIN
				INSERT INTO aircrafts (aid, aname)
					VALUES (@aircraft_id, @aircraft_name)
			END
		ELSE SET @aircraft_updated = 0;

		IF (@pilot_updated = 0 OR @aircraft_updated = 0)
			BEGIN
				INSERT INTO certified (empid, aid)
					VALUES ((SELECT empid FROM employees WHERE empid = @pilot_id),
							(SELECT aid FROM aircrafts WHERE aid = @aircraft_id))
			END
;


/*Procedures Execution*/
EXECUTE insert_pilot
	@pilot_id = 100, @pilot_lname = 'John', @pilot_fname = 'Snow',
	@aircraft_id = 900, @aircraft_name = '1903 Wright Flyer'
;

EXECUTE insert_pilot
	@pilot_id = 99, @pilot_lname = 'Daniels', @pilot_fname = 'Danielson',
	@aircraft_id = 900, @aircraft_name = '1903 Wright Flyer'
;
```

22. Δημιουργείστε έναν πυροδότη ο οποίος θα ενεργοποιείται κάθε φορά που ένας πιλότος πιστοποιείται στην λειτουργία ενός αεροσκάφους. Αν με τη νέα πιστοποίηση ο πιλότος φθάνει τις τρείς, ο πυροδότης θα αυξάνει τον μισθό του κατά 10%.

```
CREATE TRIGGER Pilot_Achievement ON certified
	AFTER INSERT AS

	BEGIN
		UPDATE employees SET salary = salary * 1.10
		WHERE empid =
		(
			SELECT c.empid FROM certified AS c
			WHERE c.empid = ANY(SELECT empid FROM INSERTED)
			GROUP BY c.empid HAVING COUNT(c.empid) = 3
		)
	END
;
```

23. Δημιουργείστε ένα πυροδότη ο οποίος θα ενεργοποιείται κάθε φορά που ενημερώνεται η τιμή του εισιτηρίου μιας πτήσης. Ο πυροδότης θα καταγράφει στον πίνακα flight_history τις παρακάτω πληροφορίες:

- Κωδικό πτήσης (fno)
- Όνομα χρήστη που έκανε την ενημέρωση
- Ημερομηνία και ώρα ενημέρωσης
- Τιμή εισιτηρίου πριν την ενημέρωση
- Τιμή εισιτηρίου μετά την ενημέρωση.

```
CREATE TABLE flight_history (
	fno VARCHAR(4) REFERENCES flights(fno),
	updated VARCHAR(500),
	date_time DATETIME,
	price_before INT,
	price_after INT
);



CREATE TRIGGER Ticket_Price_History ON flights
	AFTER UPDATE AS

	DECLARE @price_after INT,
			@price_before INT,
			@fno VARCHAR(4)

	DECLARE flight_INSERTED CURSOR FOR
		SELECT price FROM INSERTED
	DECLARE flight_DELETED CURSOR FOR
		SELECT fno, price FROM DELETED

	OPEN flight_INSERTED
	OPEN flight_DELETED

	FETCH NEXT FROM flight_INSERTED
		INTO @price_after
	FETCH NEXT FROM flight_DELETED
		INTO @fno, @price_before

	WHILE (@@FETCH_STATUS = 0)
		BEGIN
			INSERT INTO flight_history
				VALUES (
					(@fno),
					(SELECT SYSTEM_USER),
					(SYSDATETIME()),
					(@price_before),
					(@price_after)
				)

			FETCH NEXT FROM flight_INSERTED
				INTO @price_after
			FETCH NEXT FROM flight_DELETED
				INTO @fno, @price_before
		END

	CLOSE flight_INSERTED
	DEALLOCATE flight_INSERTED
	CLOSE flight_DELETED
	DEALLOCATE flight_DELETED
;
```